KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

Vera Löfberg

SEIKKAILUPELIEN KÄYTTÖLIITTYMIEN TOTEUTTAMINEN

Opinnäytetyö
Kesäkuu 2018

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Toukokuu 2018
Tietojenkäsittelyn koulutus

Tikkarinne 9
80200 JOENSUU
+358 13 260 600

Tekijä
Vera Löfberg

Nimeke
Seikkailupelien käyttöliittymien toteuttaminen

Tiivistelmä

Tässä opinnäytetyössä käsiteltiin käyttöliittymän suunnitteluun liittyviä ohjeita sekä yleisesti että seikkailupelien näkökulmasta. Tämän lisäksi opinnäytetyössä tutustuttiin työkaluihin, joilla voi toteuttaa seikkailupelin, ja tutkittiin niiden käyttöliittymän muokkausmahdollisuuksia. Aineistoa oli kerätty käyttöliittymää, videopelejä ja seikkailupelejä käsittelevistä teoksista sekä internetsivuilta.

Tämän opinnäytetyön keskeisimpiä tutkimuskysymyksiä ovat, millainen on hyvä käyttöliittymä, kuinka toteutetaan hyvä käyttöliittymä, millaiset käyttöliittymäkäytännöt ovat videopeli- ja seikkailupelikäyttöliittymissä sekä millä pelimoottoreilla voi toteuttaa seikkailupelin.

Opinnäytetyön ohella on kehitetty seikkailupelidemo, jonka käyttöliittymä on toteutettu työn käyttöliittymätietoperustan pohjalta. Pelissä on aloituskenttä sekä ensimmäinen pelikenttä. Opinnäytetyössä käytiin läpi ja tarkasteltiin peliin toteutettuja ominaisuuksia ja sitä, kuinka niihin päädyttiin. Peli on toteutettu Unity-pelimoottorilla ja C#-ohjelmointikielellä.

Käyttöliittymän selkeys ja helppokäyttöisyys ovat avainasemassa sekä videopelien että tietokoneohjelmien kehityksessä yleisemminkin, ja siihen voi myötävaikuttaa monella tapaa, kuten työstä selvisi. Pelimoottoreista yleisesti parhaimmiksi todettiin Unity ja Unreal Engine, mutta käyttäjän tarve määrittää juuri tälle täydellisen työkalun.

Kieli
suomi

Sivuja 45

Asiasanat
pelinkehitys, pelimoottori, käyttöliittymä, seikkailupeli

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
May 2018
Business Information Technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+358 13 260 600

Author
Vera Löfberg

Title
Developing User Interface for Adventure Games

Abstract

The purpose of this thesis was to address the guidelines on developing a user interface from a general point of view and from an adventure game point of view. Additionally, this thesis introduces a few tools with which it is possible to develop an adventure game and examines how well their user interface tools can be modified. The material was gathered from books and websites which deal with user interface, video games and adventure games.

The central research questions were: what is a good user interface like, how to develop a good user interface, what are the guidelines for video game and adventure game user interfaces and with which game engines can one develop an adventure game.

In addition to the thesis, there is an adventure game demo in which the user interface was developed based on the information reported in this thesis. The game includes a starter level as well as a first level. The game features and the process of how everything was decided were reported in this thesis. The game is developed with the game engine Unity and the programming language C#.

It is key for both video games and computer programs in general to develop easily understandable and easy-to-use interfaces, and there are many ways to contribute to this, as was shown in the thesis.

Language
Finnish

Pages 45

Keywords
game development, game engine, user interface, adventure game

Sisältö

1 Johdanto

Seikkailupelit ovat laaja genre, joka pohjautuu vahvasti tarinankerrontaan. Eri seikkailupelit voivat olla hyvin erilaisia ja näin ollen niiden käyttöliittymäelementit vaihtelevat paljon. Tässä opinnäytetyössä käydään läpi sekä yleisesti käyttöliittymän suunnitteluun liittyviä teorioita että ohjeita siitä, mitä videopelien, kuten seikkailupelien käyttöliittymän suunnittelussa tarvitsee pitää mielessä. Työ esittelee myös seitsemän pelimoottoria, joilla voi toteuttaa seikkailupelin ja sen käyttöliittymän. Työssä on myös käytännön osio, joka esittelee oppimani asiat käyttöliittymistä ja sisältää toteutuksen seikkailupelistä ja sen käyttöliittymästä.

Ohjelmointi on aina kiehtonut minua ja ensimmäisestä ohjelmointikurssistani tiesin, että haluan oppia aiheesta lisää. Tämän lisäksi myös videopelit ovat yksi kiinnostuksen aiheeni. Näiden faktojen takia tuntui luonnolliselta, että opinnäytetyöni käsittelee peliohjelmointia ja pelinkehitystä. Seikkailupelit ovat olleet aina lähellä sydäntäni, joten päätin ottaa sen mukaan rajoittavaksi tekijäksi. Työllä ei ole toimeksiantajaa, mutta aiheenvalinnassa on otettu se huomioon, että aiheen tutkimisesta saamani informaatio on hyödyksi tulevaisuudessani.

Millainen on hyvä käyttöliittymä? Nykypäivänä tietokone ja muut laitteet, kuten kännykät ja tabletit, ovat saaneet merkittävän aseman yhteiskunnassa ja ihmisten elämissä. Ihmiset tutkivat päivittäin laitteillaan eri internetsivuja ja lataavat uusia sovelluksia jatkuvasti. Sovelluksen tai internetsivun käyttöliittymä toimii ihmisen portaalina itse materiaaliin. Tämän takia huonosti suunniteltu ja toteutettu käyttöliittymä voi aiheuttaa hämmennystä käyttäjälle ja jopa käännyttää heidät pois käyttöliittymästä. Käyttöliittymäkokemuksen tulisi olla käyttäjälle mahdollisimman mukava ja yksinkertainen mutta myös hyödyllinen. ”A user interface is well designed when the program behaves exactly how the user thought it would.” (Spolsky 2001, 6.)

Miten sitten saadaan käyttöliittymästä käyttäjälle helpon ja miellyttävän? Hyvän käyttöliittymän salaisuus löytyy ihmisen aivoista. Ihmiset näkevät tietyt samat ominaisuudet miellyttävinä ja selvinä, ja ihmisen psykologian tutkimisen myötä on löydetty tapoja miellyttää käyttäjää luomalla aivoihin vetoavia käyttöliittymiä.

Sama pätee myös videopeleihin. Niissä pelin käyttöliittymä, johon sisältyvät sekä pelinaikaiset komponentit että tekstit kuten myös valikot, on isossa asemassa ja

koko ajan pelaajan mukana. Jos käyttöliittymä ei ole miellyttävä ja helppokäyttöinen, tämä vaikuttaa negatiivisesti pelaajan pelikokemukseen.

Opinnäytetyössä esiteltävät pelimoottorit ovat ajankohtaisia ja omasta mielestäni hyviä vaihtoehtoja seikkailupelien toteutukseen. Osa moottoreista on hyvin rajoittavia ja joilla voi vain tehdä seikkailupeli-tyyppisiä pelejä ja osa on ”vapaita” moottoreita, joissa ei peligenrekohtaisia rajoitteita ole.

2 Yleistietoa käyttöliittymistä

2.1 Käyttöliittymien historiaa

Siitä asti, kun ihmiset keksivät tietokoneohjelmia, joiden kanssa vuorovaikutus tapahtuu visuaalisen käyttöliittymän kautta, ihmiset ovat myös luoneet ohjeita ja muita sääntöjä hyvän käyttöliittymän luomiseksi. Nämä säännöt ovat kehittyneet ja muuttuneet teknologian kehittyessä ja asiantuntijoiden ymmärtäessä enemmän ihmisen psykologiasta ja siitä, mikä näyttää hyvältä käyttäjän silmään.

Muutamia mainittavia ohjeistuksien kirjoittajia vuosien varrelta ovat muun muassa Cheriton (1976), Norman (1983), Shneiderman (1987), Brown (1988) ja Nielsen ja Molich (1990). Tunnetuimpia näistä ovat Scheinerman sekä Nielsen ja Molich. Tarkemmin katsellessa ohjeistukset muistuttavat toisiaan ainakin jonkin verran. Esimerkiksi yhtenäisyys on aihe, jota painotetaan monessa ohjeistuksessa. Vaikka ohjeistukset muistuttavatkin toisiaan, jokainen niistä toi esille jotain uutta aina kehittyneen teknologian myötä. Ohjeistuksia löytyy myös eri alustoille, kuten mobiilille. (Johnson 2010, xiii.)

2.2 Hahmoteoria

Hahmoteoria eli Gestalt-teoria on kehitetty Saksassa ja Itävallassa. Se perustuu siihen, kuinka tiettyyn asiaan vaikuttaa sen ympäristö ja konteksti. “Gestalt” on saksankielinen sana, joka tarkoittaa ”kokonaisuutta” tai ”muotoa”. Teoria

keskittyy siihen, miten ihminen pystyy hahmottamaan tietyt visuaaliset muodot ja merkit. Hahmoteoria luotiin vuonna 1890. (Laine 2004.)

Yksi hahmoteorian laeista on läheisyyden laki (englanniksi “law of proximity”). Kun ihminen näkee useamman objektin lähellä toisiaan, aivot hahmottavat nämä yhtenä ryhmänä. Tämä on hyödyllistä esimerkiksi ilmoitettaessa käyttäjälle tiettyjen komponenttien käytöstä. Käyttäjä voi myös nähdä ryhmiä, vaikkei käyttöliittymän suunnittelija sitä tarkoittanutkaan, mikä voi aiheuttaa väärinymmärryksiä huonosti suunnitellussa käyttöliittymässä. (Johnson 2010, 14–16.)

Samankaltaisuuden laki (englanniksi “law of similarity”) tarkoittaa sitä, että ihminen näkee samankaltaiset objektit omana ryhmänään. Aivot pystyvät hahmottamaan samankaltaiset muodot muiden muotojen joukosta, mikä helpottaa eri käyttöliittymäobjektien erittelyä. (Johnson 2010, 16–17.)

Ihmisaivot pystyvät jatkuvuuden lain (englanniksi “law of continuity”) mukaan näkemään kokonaisen kuvion, vaikka se ei olisikaan kokonainen. Esimerkiksi kaksi peräkkäistä viivaa, jotka ovat yhdensuuntaiset, voivat silti näyttää ihmiselle yhdistetyltä, vaikka ne eivät koskettaisikaan toisiaan. Toinen hyvä esimerkki on IBM:n logo, joka koostuu vaakasuorista viivoista, jotka eivät yhdisty, mutta ihminen pystyy silti hahmottamaan kirjaimet ”IBM”. (Johnson 2010, 18.)

Samaan tapaan sulkeutuvuuden lain (englanniksi “law of closure”) mukaan ihminen pystyy näkemään kuvion, vaikka se olisi vajaa tai keskeneräinen. Ihminen pystyy siis esimerkiksi hahmottamaan viivoista, että kyseessä on ympyrä, vaikka osa ympyrän viivoista ei näy. Jos kyseinen keskeneräinen kuvio hahmotetaan suljettuna, ihminen näkee tämän omana objektinaan tai alueena. (Johnson 2010, 19.)

Kun ihminen näkee kuvion, jonka tarkoitus ei ole täysin selvä, hänen aivonsa yrittävät selventää sitä etsimällä siitä symmetrisiä muotoja. Tätä kutsutaan symmetrian laiksi (englanniksi “law of symmetry”). Symmetria yleensä myös kiinnittää ihmisen huomion sen miellyttävyyden vuoksi. (Johnson 2010, 20.)

Kohde ja alusta -laki (englanniksi “law of figure and ground”) tarkoittaa sitä, että ihminen jakaa kuvassa olevat objektit joko edessä oleviksi kohteiksi tai enemmän takana oleviksi alustoiksi. Kohteet ovat nimensä mukaisesti ihmisen katseen kohteena, kun taas alustat toimivat enemmän taustoina. Kohteet ja alustat voivat myös vaihtaa rooleja riippuen siitä, mihin käyttäjän katse ohjataan. Joissain

tilanteissa kohteena voi olla esimerkiksi pop-up-viesti, jolloin kaikki muu taustalla oleva tummennetaan hieman, eli kohteesta tulee alusta. (Johnson 2010, 21–23.)

Jos useampi objekti liikkuu samalla tavalla, kuten edestakaisin samaan aikaan, nämä objektit hahmottuvat ihmisille omana ryhmänä. Tämä on yhteisen liikkeen laki (englanniksi “law of common fate”). Objektien ei myöskään tarvitse olla samassa paikassa, vaan samanlainen liike riittää yhteyden huomaamiseen. (Johnson 2010)

Yhdistämisen laki (englanniksi “combined”) on ylempänä mainittujen hahmolakien yhdistämistä. Yleensä lakeja ei hyödynnetä yksitellen vaan nimenomaan yhdistettynä, jolloin ne luovat käyttäjän silmää miellyttävän käyttöliittymän. Näitä hahmolakeja yhdistettäessä on hyvä pitää silmällä, ettei niitä tule tahattomasti käyttäneeksi yhteyksissä, joissa niiden ei ole tarkoitus olla. Tällainen voi hämätä käyttäjää ja aiheuttaa väärinkäsityksiä ja ohjelman väärinkäyttöä. (Johnson 2010, 25–27.)

2.3 Vältettäviä asioita

Hahmoteoria käsittelee sääntöjä, jotka kannattaa pitää mielessä, jotta saa aikaan hyvän käyttöliittymän. Käyttöliittymää tehdessä on myös hyvä välillä katsoa toiseen suuntaan eli sitä, mitä ei kannata tehdä.

Jos käyttöliittymä antaa paljon sekalaista informaatiota yhdellä kertaa, käyttäjä voi häkeltyä. Informaation tulisi olla esillä selkeästi ja esimerkiksi lyhyempinä pätkinä, jotta käyttäjän ei tarvitsisi käyttää runsaasti aikaa tekstin selvittämiseen. Itse käyttäjä on se, jolle suunniteltu tuote tai internet-sivu on kohdistettu, joten käyttäjän mukavuus tulisi olla ensimmäisenä mielessä. (Siang 2018.)

Erilaisissa valikoissa ja sivustoissa navigoinnin tulisi olla helppoa käyttäjälle. Jos navigointi tapahtuu valikosta, jossa on pelkästään kummallisen näköisiä kuvakkeita, käyttäjän täytyy arvata, mitä mistäkin tapahtuu ja miten hän pääsee halutuun kohteeseen. Tämän voi estää antamalla navigointipainikkeille ja muille painikkeille selkeät kuvaukset, joista ei voi erehtyä. (Siang 2018.)

Animaatiot ovat loistava tapa elävöittää esimerkiksi internet-sivustoa, mutta huono animaatio voi olla ennemminkin turhauttava. Hyvällä animaatiolla on

tarkoitus, esimerkiksi tiimalasikuvakkeen ilmestyminen tiettyyn objektiin, kun ohjelma tai sivusto tallentaa syötettyjä tietoja, ja tiimalasin katoaminen, kun tallennus on valmis. (Siang 2018.)

Myös responsiivisuus on tärkeää käyttöliittymässä. Käyttöliittymän tulisi pystyä dynaamisesti muuttaa kokoaan ja sijaintiaan käyttäjän mukaisesti. Responsiivisuuteen tulee kiinnittää huomiota varsinkin, jos käyttöliittymää käytetään useammalla erilaisella laitteella, kuten tietokoneella ja tabletilla. (Sarah 2017.)

Käyttöliittymän eri osien tulisi aina olla yhtenäisiä ja kontekstiin sopivia. Epäyhtenäinen käyttöliittymä sotii käyttäjän silmää vastaan ja tekee käyttöliittymäkokemuksesta epämiellyttävämmän. (Sarah 2017.)

2.4 Värit

Värivalinoilla on suuri merkitys. Värit ovat koko ajan näkyvillämme, ja ne myös vaikuttavat mielipiteisiimme eri asioista. Näin ollen ne ovat myös kriittinen osa käyttöliittymäsuunnittelua. Liian tasaiset ja saman sävyiset värit eivät ohjaa käyttäjää tarpeeksi eivätkä myöskään tee vaikutusta. Toisaalta liian vahvat värit, jotka sotivat toisiaan vastaan, kuten vahva punainen ja sininen, ärsyttävät liikaa käyttäjän silmää. Käytettävien värien tulisi sopia hyvin yhteen ja antaa käyttäjälle miellyttävä kokemus.

Käyttöliittymä voi olla hyvin uuvuttavan näköinen tai se voi olla vaikea hahmottaa, jos siinä ei ole kontrastia. Jos sekä taustat että tekstit ovat hyvin saman värisiä tai sävyisiä, käyttäjän on vaikeampaa hahmottaa, minne hänen tulee kiinnittää huomiota. (Sarah 2017.) Pienet sävyn tai värin muutokset voivat olla liian huomaamattomia, joten selkeästi erilaisen värin käyttäminen on paljon parempi vaihtoehto. Hyvä tapa testata värejä on muuttaa ne harmaansävyisiksi. Näin voi helposti tarkastella, ovatko valitut värit tarpeeksi erilaisia. (Johnson 2010, 37.)

Kontrastin lisäksi tietyt värit saavat helposti käyttäjän huomion. Nämä värit ovat musta, valkoinen, punainen, vihreä, keltainen ja sininen. Eri värien käyttö lisää selkeyttä käyttöliittymässä. Värejä tulee kuitenkin käyttää varovasti. Varsinkin vastavärejä käyttäessä lopputulos voi olla epämiellyttävä käyttäjälle. (Johnson 2010, 37.)

Mainitsemisen arvoista on myös se, että käyttäjien joukossa voi olla värisokeita ihmisiä, jotka eivät välttämättä pysty erottamaan tiettyjä värejä. Käyttöliittymien suunnittelijoiden tueksi on olemassa paljon materiaalia, jonka avulla näitä tiettyjä väriyhdistelmiä pystytään välttämään. Tällaisia väriyhdistelmiä ovat muun muassa tummanpunainen ja musta, tummanpunainen ja tummanvihreä, sininen ja violetti sekä vaalean vihreä ja valkoinen. (Johnson 2010, 37.)

2.5 Videopelien käyttöliittymät

2.5.1 Yleiset videopelikäyttöliittymän säännöt

Videopelikäyttöliittymää tehdessä olisi hyvä pitää mielessä tietyt säännöt, jotta lopputulos ei vahingoita pelaajan pelikokemusta. Mitchell mainitsee tärkeinä käyttöliittymäohjeina mm. nämä kolme asiaa; 1) käyttöliittymä täytyy pitää mahdollisimman selkeänä ja yksinkertaisena, 2) itse käyttöliittymän osuus ei saisi viedä peliruudulta liikaa tilaa ja 3) käyttöliittymän ulkoasun tulisi olla yhteensopiva pelin kanssa. Käyttöliittymän yksinkertaisuus on tärkeää, jotta pelaaja pääsee haluttuihin ominaisuuksiin ja tietoihin nopeasti käsiksi kesken pelaamisen. Samoin käyttöliittymän liian suuri koko voi heikentää pelaajan keskittymistä ja kiinnostusta itse peliin. (Mitchell 2012.) Kuten Lecky-Thompson (2008, 209) hyvin totesi, ”It is no good using Star Trek -style technology in Mad Max and the Thunderdome”.

2.5.2 Yleisimmät pelikäyttöliittymien komponentit

HUD eli “Heads-Up Display” on yksi keskeisimmistä käyttöliittymän osista. HUD antaa informaatiota pelaajan sen hetkisestä tilasta. Se kehitettiin alun perin hävittäjälentäjille, joiden piti saada nähtyä tarvittava informaatio mahdollisimman nopeasti mutta myös samalla pitää katse menosuunnassa ja sen hetkisessä tilanteessa. Koneessa HUD näkyi lentäjien kypärän visiirissä tai koneen tuulilasissa, ja se antoi lentäjille tarvittavat tiedot ilman, että lentäjän tarvitsi liikuttaa päätään ja katsoa poispäin. Peleissä HUD antaa tietoja, kuten onko pelaajan tarpeellista parantaa hahmoaan, pelaajan ja hahmon eri pisteet, kuluneen ajan,

pelaajan käytössä olevat aseet ja niiden jäljellä olevat ammukset ja kaikki muut tarpeelliset tiedot. (Mitchell 2012, 144.)

Heti pelin käynnistyessä pelaajaa tervehtii pelin päävalikko. Se voi olla pelaajan ensimmäinen vuorovaikutus pelin kanssa, joten ensivaikutelma on tärkeä. Päävalikko koostuu yleensä taustakuvasta, joka liittyy peliin, ja eri painikkeista, kuten ”Aloita peli” ja ”Asetukset”. Pelistä riippuen vaihtoehtoja voi olla monta. Päävalikon tulisi olla mahdollisimman selkeä ja saada pelaaja kiinnostuneeksi pelistä.

Päävalikon lisäksi yksi tarpeellinen komponentti on taukovalikko. Tämä on valikko, joka ilmestyy, kun pelaaja haluaa pysäyttää pelin hetkellisesti. Taukovalikon tarkoitus on se, että pelaaja voi pitää pelistä tauon tai että pelaaja pääsee käsiksi tiettyihin toimintoihin taukovalikossa. Näitä toimintoja voivat olla asetukset, pelin päävalikkoon palaaminen ja muut toiminnot pelistä riippuen, kuten pelitason uudelleenaloitus.

2.5.3 Pelien käyttöliittymien värit

Yksi universaaleimmista asioista kaikessa suunnittelussa ovat värit, kuten alaluvussa 2.4 tuli jo esille. Peleissä värejä käytetään pelihahmon elämämittarista pelin taustan piirtoon, ja yleensä näillä on omat merkityksensä.

Punainen on universaalisti vaaran merkki, ja sitä pidetään hyökkäävämpänä kuin muita päävärejä. Keltainen ja vihreä taas ovat enemmän rohkaisevia värejä ja vähemmän uhkaavia. Sininen on yleensä neutraali. Esimerkiksi pelaajahahmon elämämittari voi käyttää näitä värejä. Jos mittari on vihreä tai keltainen, pelaaja tietää ilman itse tekstin lukemista, että tällä hetkellä on kaikki hyvin. Sen sijaan jos mittari on punaisella, se yleensä tarkoittaa sitä, että pelaajan kannattaa olla varovainen. Samoin taistelussa punaisella värillä yleensä kuvataan vihollista tai jotain muuta hyökkäävää olentoa. (Lecky-Thompson 2008, 194–195.)

2.5.4 Diegeettinen teoria

Videopelikäyttöliittymän suunnittelussa on hyvä tutustua diegeettiseen teoriaan, joka jakaa käyttöliittymät neljään eri kategoriaan sen perusteella, ovatko ne osa pelimaailmaa ja ovatko ne mukana tarinankerronnassa.

Ei-diegeettinen käyttöliittymä tarkoittaa käyttöliittymää, joka ei ole tarinankerronnassa mukana ja ei ole olemassa itse pelimaailmassa vaan näkyy vain pelaajalle yleensä 2D-kuvin ja -tekstinä. Tämä on yksi yleisimmistä käyttöliittymistä, jossa muun muassa pelaajan status ja muut tiedot näkyvät esimerkiksi ruudun reunoilla koko ajan pelaajalle. Tämä ei tosin ole ainut tapa näyttää pelaajan ja ympäristön tietoja. (Russell 2011)

Ei-diegeettisen vastakohta, **diegeettinen käyttöliittymä**, taas on elementtinä tarinankerronnassa ja on osana itse pelimaailmaa eikä näy vain pelaajalle. Esimerkiksi jokin pelaajahahmon omistama laite tai esine voi toimia diegeettisenä käyttöliittymänä. Tätä käyttöliittymää käytetään varsinkin scifi-genren peleissä, sillä tulevaisuuteen on hyvä suunnitella ”high-tech”-laitteistoa, vaikkei sellaista nykymaailmassa vielä olisikaan olemassa. Teknologia kehittyy koko ajan, ja maailmalla on esimerkiksi nyt jo puettavaa tai päällä pidettävää teknologiaa, kuten Apple Watch ja erilaiset aktiivisuusrannekkeet. Hyvänä esimerkkinä ovat myös Under Armour -valmistajan Gemini 3 RE -älykengät, jotka pitävät kirjaa käyttäjän juoksemisesta ja myös analysoivat nämä tiedot ja niiden perusteella antavat ehdotuksia käyttäjälle. Tätä teknologian kehitystä voidaan käyttää inspiraationa varsinkin scifi-peleissä, jossa voidaan kuvitella, että teknologia on kehittynyt vieläkin pidemmälle, kuten älypaitoihin tai hologrammeihin. Diegeettistä käyttöliittymää on tosin hieman hankala yrittää integroida esimerkiksi muinaisen Rooman aikaan sijoittuviin peleihin, ja tämän takia sitä esintyykin enemmän scifi-genren peleissä. (Russell 2011)

Jos ei-diegeettisessä käyttöliittymässä HUD ei ole mukana tarinankerronnassa eikä ole pelimaailmassa ja diegeettisessä taas on, niin mitä välivaihtoehtoja löytyy? **Avaruudellinen käyttöliittymä** on olemassa pelimaailmassa, mutta se ei näy pelihahmoille eikä ole osa tarinankerrontaa. Tämä toimii, jos pelissä on muun muassa tiettyjä esineitä, joiden kanssa pelaaja voi olla vuorovaikutuksessa. Vuorovaikutteisten esineiden ympärille voi tulla erivärinen korostus tai jokin muu

efekti, kuten kimalleanimaatio. Pelihahmo ja pelimaailma eivät reagoi tähän mitenkään, mutta tätä kautta pelaaja voi saada informaatiota. Esimerkki pelistä, jossa tätä käytetään, on The Sims -pelisarja.

Avaruudellisen vastakohtainen käyttöliittymä on **metakäyttöliittymä**, joka ei ole olemassa pelimaailmassa mutta jota käytetään tarinan kertomisessa. Esimerkiksi jos pelaajahahmo loukkaantuu pahasti, pelaajan ruudulle voi ilmestyä hetkellisesti veriroiskeita tai ruutu voi mennä punaiseksi tai mustavalkoiseksi, kunnes pelihahmon haava paranee. (Russell 2011)

2.5.5 Seikkailupelien käyttöliittymäelementit

Seikkailupelit ovat hyvin monipuolinen ja laaja genre, mutta niissäkin löytyy keskenään samanlaisia UI-elementtejä. Kuten muissakin peleissä, myös seikkailupeleissä voi olla pelaajahahmon health bar ja muut pelaajahahmon ominaisuudet, kuten mana ja stamina (kuva 1). Moni seikkailupelien UI-elementeistä liittyy myös esimerkiksi matkustamiseen tai sijaintiin.

Kuva 1. Kuva CaveStory+ -pelin healthbarista (punainen mittari) ja valitun aseen kokemuspisteistä (keltainen mittari) (Kuva: Vera Löfberg)

Yksi sijaintiin liittyvistä elementeistä on kartta. Jos pelaajalla on mahdollisuus tutkia isompaa maailmaa tai maailmaa, jossa on useampi sijainti, pelissä on

mahdollisesti kartta, jonka avulla pelaaja pystyy paikantamaan itsensä suhteessa maailmaan (kuva 2). Kartta voi yleensä olla valittavana jollain tietyllä painikkeella, mutta pelissä voi myös koko ajan olla näkyvissä minikartta. Minikartta on yleensä pelinäkymän reunassa, ja se näyttää pelaajalle tämän sijainnin ja lähiympäristön. Näin pelaajan ei tarvitse avata kokonaista karttaa aina kun hän haluaa tietää, onko menossa oikeaan suuntaan.

Kuva 2. Jotun-pelissä koko pelialueen kartta on integroitu osaksi peliä ison kivimaalauksen/-kaiverruksen lailla. (Kuva: Vera Löfberg)

Toinen matkustamiseen liittyvä elementti on kompassi. Kompassi on käytössä reaalimaailmassa suunnan löytämisessä, ja samoin myös seikkailupelit ovat ottaneet sen mukaan avuksi suunnistamisessa. Kompassi voi olla oikeannäköinen kompassi, jonka pelaajahahmo ottaa esille välillä ja suunnistaa sen mukaan. Näin ollen kompassi on osa diegeettistä käyttöliittymää (ks. alaluku 2.5.4). Kompassi voi olla myös ruudulla näkyvä viivanmuotoinen objekti, joka näyttää ilmansuuntaan, johon pelaaja sillä hetkellä katsoo. Tällä kompassilla voi myös nähdä lähellä olevat paikat, lähellä olevat tehtävät ja muuta informaatiota. Tällainen kompassi voi myös korvata minikartan. Kompasseja löytyy esimerkiksi seikkailupeleistä Firewatch ja Stranded Deep, joissa kummassakin pelaaja voi käyttää sitä apunaan suunnistuksessa.

Tarinankerronta seikkailupeleissä on melko tärkeää, joten se, mitä pelaajan pitää tehdä seuraavaksi tarinassa, tulee tehdä selväksi pelaajalle. Yksi tapa tähän on näyttää pelaajan sen hetkinen tavoite. Aina kun pelaajan tavoite on sitten saavutettu, seuraava tavoite pitäisi tehdä selväksi. Tavoitteen päivitysvaiheessa tämän voi esimerkiksi esittää pelaajalle tekstinä. Muuna aikana tavoitteen tarkastaminen voi olla napin painalluksen päässä, tai tavoite voi olla näkyvillä koko ajan käyttöliittymässä. Submerged-pelissä, pelaajahahmo voi todeta seuraavan tehtävän itsekseen ääneen, esimerkiksi että ”nyt on nälkä, täytyy etsiä ruokaa”.

Seikkailupeleissä, kuten peleissä yleisestikin, inventaario on melko yleinen. Varsinkin seikkailupeleissä voi olla tarvetta ratkaista pulmia eri esineiden avulla tai muuten vain löytää tavaraa, jonka pelaajahahmo ottaa mukaansa. Nämä tavarat menevät inventaarioon. Inventaarion pystyy yleensä avaamaan tietyllä painikkeella, mutta osa inventaariosta voi myös näkyä koko ajan ruudulla. Jotkut pelit toimivat niin, että inventaariosta valitaan yksi esine, joka on sillä hetkellä käytössä. Esimerkiksi jos pelaaja haluaa avata lukitun oven, inventaariosta valitaan avain aktiiviseksi esineeksi, ja seuraavan kerran kun pelaaja vuorovaikuttaa oven kanssa, sen saa avattua (kuva 3). Tämä aktiivinen esine näkyy ruudulla, jotta pelaaja muistaisi sen olevan aktiivinen.

Kuva 3. OneShot -pelin inventaario ja aktiivinen esine. (Kuva: Vera Löfberg)

Jos pelissä on tulitaistelua, on tärkeää näyttää inventaarion lisäksi myös aktiivinen ase. Yleensä pelaajan valitsema ase näkyy ruudun reunassa joko nimenä tai kuvana. Valitun aseen lisäksi käyttöliittymässä näytetään aseeseen sopivat luodit

ja niiden lukumäärä pelaajan inventaariossa (kuva 4). Näin pelaaja saa myös tiedon siitä, onko tarvetta vaihtaa asetta tai vaikka lähteä karkuun taistelusta.

Kuva 4. Hyper Light Drifterissä pelaajan luodit näkyvät pinkkeinä palikoina ruudun vasemmassa yläkulmassa. Näitä palikoita saa lisää sivaltamalla vihollisia miekalla. (Kuva: Vera Löfberg)

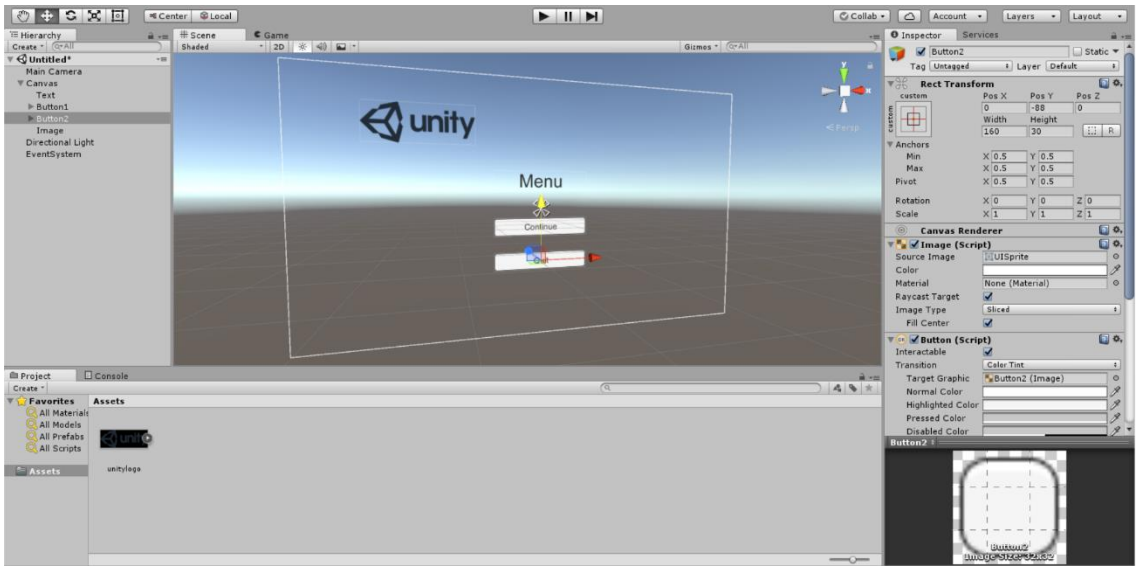
3 Eri pelimoottorien käyttöliittymätyökalut

3.1 Unity

3.1.1 Yleisesti

Unity 3D -pelimoottori on yksi tunnetuimmista pelimoottoreista pelialalla. Unityn omistaa Unity Technologies, joka kehitti Unityn ensimmäisen version vuonna 2005. Tämän jälkeen Unity Technologies on julkaissut kuusi pääversiota. Tällä

hetkellä Unitylla voi kehittää pelejä 19 alustalle ja sen lisäksi yhdeksälle eri virtual reality -alustalle. Nimestään huolimatta Unity tukee sekä 3D- että 2D-pelien tekemistä. Yksinkertaisia pelejä on mahdollista toteuttaa ilman ohjelmointia esimerkiksi käyttämällä valmiita frameworkeja, mutta kuten monessa muussakin pelimoottorissa, skriptaus antaa kehittäjälle monipuolisuutta ja enemmän vapautta. Tämänhetkisessä versiossa skriptaus tapahtuu C#-kielellä ja JavaScriptilla. Unity tuki aiemmin myös Boo-ohjelmointikieltä, mutta se vanhentui aikaisemmissa versioissa. Myös JavaScript on poistumassa uusimmista versioista. Pelien tekijöiden kehityksen avuksi Unitylla on oma Asset Store, josta voi ostaa tai ladata ilmaiseksi erilaisia resursseja, kuten frameworkeja ja 3D-malleja. (Wikipedia: Unity 2018.)



Kuva 5. Yksinkertainen Unitylla tehty käyttöliittymä (Kuva: Vera Löfberg)

3.1.2 Unityn käyttöliittymän toiminta

Kaikki Unityn käyttöliittymäobjektit eli UI-objektit pohjautuvat siihen, että jokainen objekti on vanhempiobjektin lapsi ja kaikkien UI-objektien päävanhempana on Canvas-objekti (kuva 5). Ilman Canvas-objektia käyttöliittymä ei näy pelissä. Jos projektiin liittää UI-objektin ilman että projektissa on Canvas-objektia, Unity lisää Canvas-objektin automaattisesti. Canvas-objekteja voi olla useampia, jos on tarve. Canvas-objektissa on kiinnitettynä Canvas-komponentti, joka määrittää

valitun Canvas-objektin asetukset. Komponentin Render Mode -asetus määrittää, renderöidäänkö Canvas-objekti suoraan näytölle 2D-objektina vai pelimaailmaan normaalina 3D-objektina. (Unity 3D 2014).

Render Mode -asetus tarjoaa kaksi eri vaihtoehtoa näytölle renderöimiseen, jotka ovat Overlay ja Camera. Overlayssa renderöinti tapahtuu suoraan peliskenen päälle, ja Camerassa se tapahtuu kiinnitettynä valittuun kameraan. Näytölle renderöidessä Canvas-objektin kokoa ei voi muuttaa, vaan se päivittää itsensä näytön kokoiseksi. Pelimaailmassa renderöitynä Canvas-objektin kokoa voi muuttaa, ja sitä kohdellaan enemmän kuten normaalia 3D-objektia. Eri UI-objektien renderöintijärjestys määräytyy skenen objektien hierarkian mukaan, eli viimeinen objekti renderöidään viimeisenä ja se tulee muiden päälle. (Unity 3D 2014).

Objekteissa on oletuksena melkein aina mukana Transform-komponentti, joka määrittää peliobjektin sijainnin ympäristössä. Transform-komponentin sijasta UI-objekteissa on RectTransform-komponentti. Näiden kahden erona on lähinnä vain RectTransformin käyttämät ankkurit, joilla saa sidottua UI-objektin tietylle paikalle. Unityssa on useampi valmis ankkuripaikka valmiina käyttöön, tai käyttäjä voi myös määrittää halutun ankkurin itse. (Unity 3D 2014).

3.2 Unreal Engine

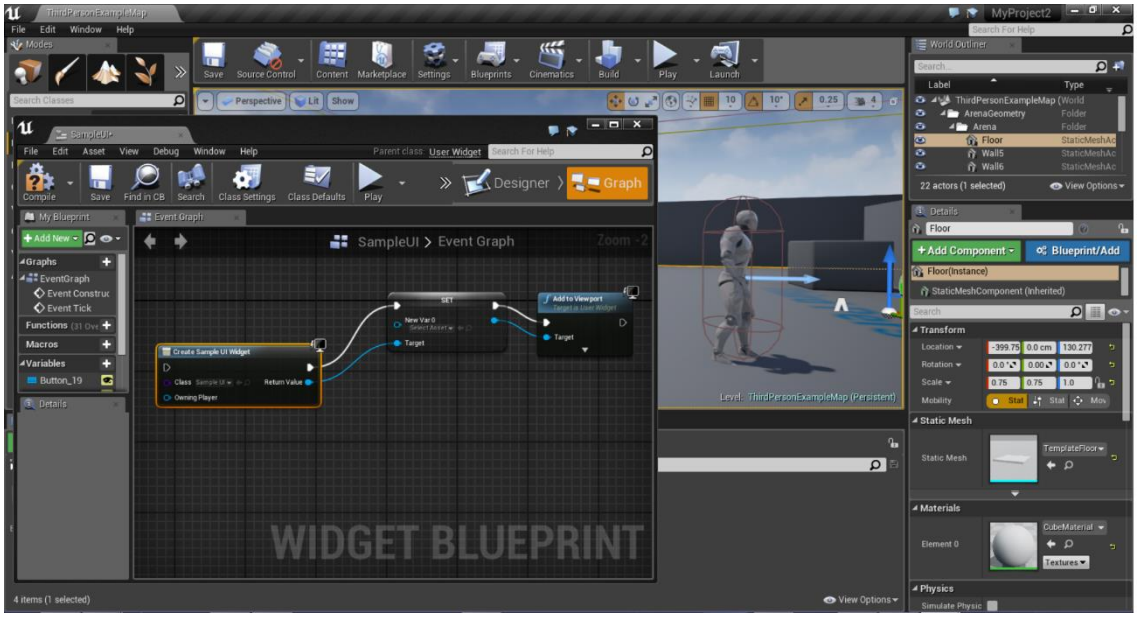
3.2.1 Yleisesti

Toinen tunnettu pelimoottori pelialalla on Unreal Engine. Unreal Enginen omistaa Epic Games, ja se julkaistiin ensimmäisen kerran vuonna 1998. Unreal Enginestä on ensimmäisen julkaisun jälkeen tullut kolme versiota lisää, ja nykyinen versio Unreal Engine 4 mahdollistaa pelien tekemisen kymmenelle alustalle ja kuudelle virtual reality -alustalle. Unreal Engine on suunniteltu 3D-pelejä varten, mutta myös 2D-pelin tekeminen on mahdollista tässä kehitysympäristössä. Unreal Enginellä voi tehdä pelejä ilman koodia, mutta ympäristössä on myös mahdollista ohjelmoida. Skriptaus tapahtuu C++-kielellä. Unreal Enginessä oli aiemmin käytössä UnrealScript (lyhenne UScript), mutta vuonna 2012 Epic Games päätti, että C++ on parempi vaihtoehto. Unreal Enginellä on oma kauppapaikka, josta

kehittäjät voivat ostaa tai ladata ilmaiseksi muiden ihmisten tekemiä resursseja. (Wikipedia: Unreal Engine 2018).

3.2.2 Unreal Motion Graphics UI

Unreal Engine 4:ssä käyttöliittymien luonti tapahtuu Unreal Motion Graphics UI Designer -työkalun (lyhenne UMG) kautta. UMG toimii niin, että jokainen UI-elementti koostuu widgeteistä, ja widgettejä muokataan Widget Blueprint -editorissa. Widget Blueprint on visuaalinen käyttöliittymä eri pelikäyttöliittymien luomiseen ja niiden toimintojen toteuttamiseen. Editorissa on kaksi näkymää, joista toisessa muokataan UI-objektin visuaalista ulkoasua (kuva 6) (englanniksi “designer”) ja toisessa objektin toiminnallisuutta (englanniksi “graph”). (Unreal Engine 2018).



Kuva 6. Unreal Enginen Designer-näkymä. (Kuva: Vera Löfberg)

Designer-näkymässä voi muokata UI-objektista haluamansa näköisen hiirellä muokkaamalla ja eri asetuksia vaihtamalla. Näkymässä on mukana Palette, josta löytyy kaikki valmiit käytettävät widget-komponentit eri kategorioihin lajiteltuna. Kategoriat ovat Common, Extra, Input, Optimization, Panel, Primitive, Special Effects, Uncategorized ja User Created; viimeksi mainitussa on käyttäjän itse

luomat widgetit. Palettesta voidaan lisätä widgettejä raahaa ja pudota -tyylillä hiirellä vetämällä ne Blueprint Editoriin. Designer-näkymän kautta voi myös animoida esimerkiksi painikkeen painamisen eri vaiheet ja muut animaatiot. Animaatiot määritellään Animation Trackissa, ja jokaisella widgetillä voi olla useampi Animation Track, mikä mahdollistaa useamman animaation käytön yhdessä widgetissä. Animaatiomahdollisuuksia ovat esimerkiksi widgetin värin muuttaminen tai sen sijainnin, skaalauksen ja rotaation muuttaminen. Eri widgeteillä voi olla hieman erilaiset animaatiomahdollisuudet. (Unreal Engine 2018).

Kaikkien painikkeiden ja muiden vuorovaikutetteisten käyttöliittymäobjektien toiminnallisuudet tehdään Blueprint Editorin Graph-näkymässä. Päänäkymään muodostetaan ajatuskarttamainen rakenne eri widgeteistä, funktioista ja niiden yhteyksistä. Tämä on visuaalista skriptausta, joka on sopivaa myös ohjelmoinnin aloittelijalle. Graph-näkymässä käyttäjä voi luoda omat funktionsa ja määrittää, miten kaikki toimii. (Unreal Engine 2018).

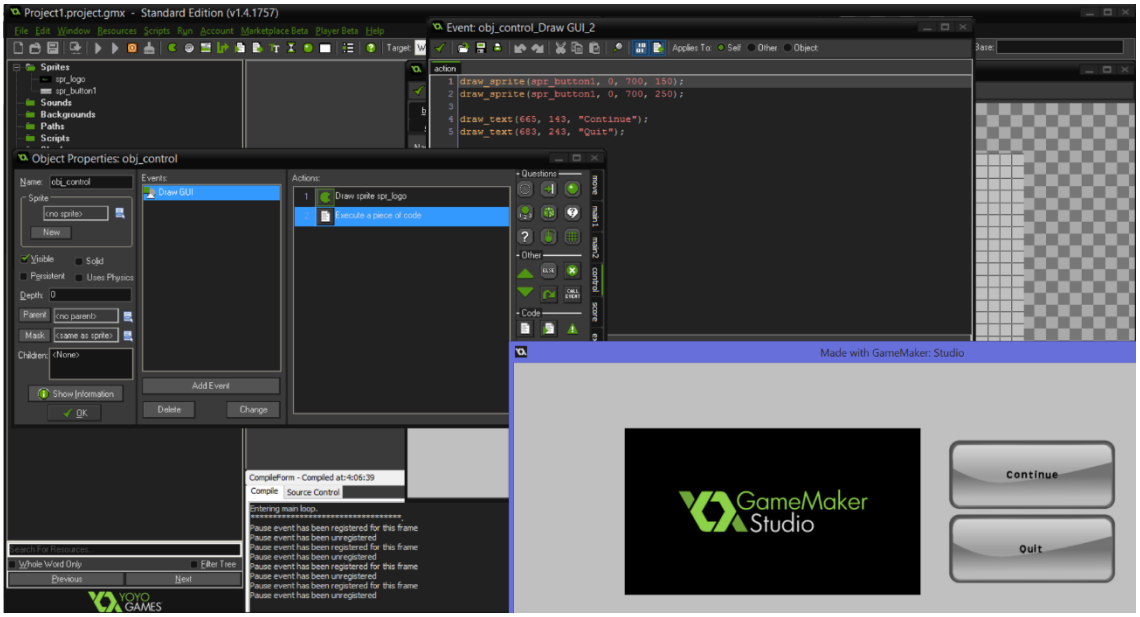
3.3 GameMaker Studio

3.3.1 Yleisesti

GameMaker Studio on hieman ominaisuuksiltaan suppeampi kuin aiemmin mainitut pelimoottorit mutta silti käytännöllinen ja hyvä vaihtoehto pelinkehittämiseen. GameMaker Studion ensimmäinen versio julkaistiin vuonna 1999, ja sen kehittäjä on YoYo Games. Tällä hetkellä GameMaker Studiosta on kahdeksan versiota, joista uusin on GameMaker Studio 2.1. GameMaker Studiolla voi kehittää pelejä 11 eri alustalle. Pelimoottorilla tehdään pääasiassa 2D-pelejä, mutta on myös mahdollista käyttää 3D-grafiikoita; muut 3D-toiminnallisuudet jäävät tosin vajaiksi. GameMaker Studiolla on mahdollista kirjoittaa skriptejä sen omalla ohjelmointikielellä, GameMaker Languagella (lyhenne GML). Kehittäjien tukena on myös YoYo Gamesin Marketplace, jossa on tarjolla muiden käyttäjien tekemiä resursseja. (Wikipedia: GameMaker Studio 2018).

3.3.2 GameMaker Studion käyttöliittymätoiminnot

GameMaker Studion toiminnallisuudet toimivat tapahtumien (englanniksi “event”) avulla, jotka ovat kiinni objekteissa. Käyttöliittymän tapauksessa käyttäjä voi tehdä esimerkiksi objektin nimeltä obj_GUI ja liittää tähän objektiin GameMaker Studion tapahtumat Draw GUI Begin, Draw GUI ja Draw GUI End. ”This means that you can have an instance draw all the elements for your HUD without having to base all the positioning on the position within the room of the instance and the position of the current view either” (YoYoGames Ltd. 2018). Draw GUI -tapahtuman sisälle tulee kaikki, mitä näytölle piirretään ja milloin. Tapahtumassa käytettäviin piirtotoimintoihin sisältyy mm. Draw Sprite, Draw Background, Draw Text, Draw Scaled Text, Draw Rectangle, Horizontal ja Vertical Gradient, Draw Ellipse, Gradient Ellipse, Draw Line, Draw Arrow, Draw Button, Draw Set Circle Precision, Draw Healthbar ja Draw Path. (YoYoGames Documentation 2018).



Kuva 7. GameMakerin raahaa ja pudota -järjestelmä ja Execute Script (Kuva: Vera Löfberg)

GameMaker Studiossa on mahdollista kehittää pelejä ilman ohjelmointia sen raahaa ja pudota -tyylisen tapahtumien ja toimintojen liittämisen ansiosta (kuva 7).

Objekteihin voi kuitenkin lisätä Execute Script -toiminnon, johon voi kirjoittaa omaa koodia GameMaker Languagella.

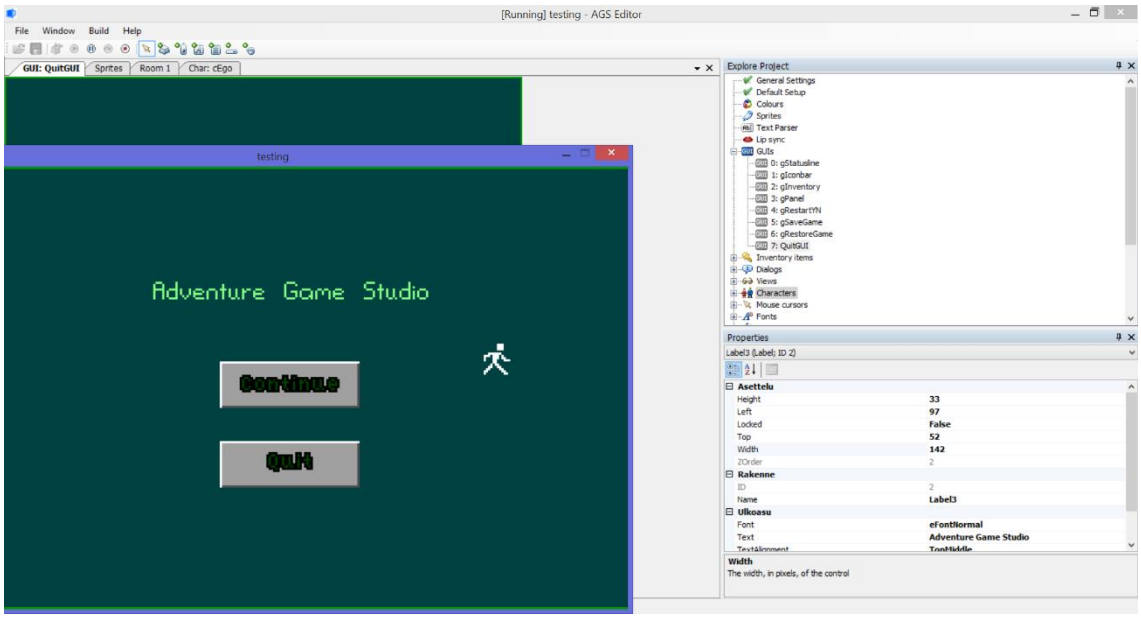
3.4 Adventure Game Studio

3.4.1 Yleisesti

Adventure Game Studio (lyhenne AGS) on pelinkehitysympäristö, joka on suunniteltu nimenomaan 2D-seikkailupelien tekemiseen. Adventure Game Studion kehittäjä on Chris Jones, ja sen ensimmäinen versio julkaistiin vuonna 1997 nimellä Adventure Creator. Tällä hetkellä Adventure Game Studiosta on olemassa kolme versiota. Vaikka ohjelmiston nimi muistuttaa hieman aikaisemmin mainittua GameMaker Studiota, näillä ohjelmistoilla ei ole mitään tekemistä toistensa kanssa. Adventure Game Studiossa voi tehdä pelejä ilman ohjelmointia tai sitten voi käyttää sen omaa AGS-script-ohjelmointikieltä toiminnallisuuksien luomiseen. (Wikipedia: Adventure Game Studio 2018).

3.4.2 Adventure Game Studion käyttöliittymätoiminnot

Adventure Game Studion käyttöliittymä toimii piirtämällä eri GUI-objekteja päällekkäin taustan päälle (kuva 8). GUI-objekteja on projektissa oletuksena valmiina kaksi: Statusline ja Icon Bar. Näitä ei ole kuitenkaan projektissa pakko käyttää. GUI on Adventure Game Studiossa suorakulmion muotoinen objekti. Jokaisella GUI-objektilla on kolme eri näkyvyysasetusta: 1) aina näkyvissä, 2) näkyvissä hiiren ollessa tietyllä alueella näytöllä ja 3) näkyvissä koodin aktivoimana. (Adventure Game Studio Manual 2018).



Kuva 8. Esimerkki Adventure Game Studion GUI-objekteista (Kuva: Vera Löfberg)

Adventure Game Studiossa on valmiina mallina käytettävissä inventaariojärjestelmä. Inventaario on aina kiinni pelihahmossa. Jos pelissä on useampi hahmo, joilla ei ole käytössä samat esineet, jokaisella hahmolla tulee olla oma inventaario. Oletusinventaario vaatii jokaisen inventaarioon laitettavan esineen grafiikan ja tiedon siitä, kuinka esineen kanssa voi vuorovaikuttaa. Käyttäjällä on myös mahdollisuus toteuttaa oma kustomoitu inventaario Adventure Game Studiossa. Animaatiomahdollisuudet käyttöliittymäpuolella ovat suppeat, mutta objektit ja taustat on mahdollista animoida. Skriptauksen avulla pääsee myös käsiksi monipuolisemmin eri GUI-funktioihin. Muita käytössä olevia GUI-elementtejä ovat GUI-painikkeet, -tekstit, -teksti-ikkunat, -liukusäätimet, -tekstilaatikot ja -luettelolaatikot. (Adventure Game Studio Manual 2018).

3.5 CryEngine

3.5.1 Yleisesti

CryEngine on vuonna 2002 kehitetty pelimoottori, jonka omistaa CryTek. CryEngine on tunnettu Far Cry -sarjan peleistä, jotka on tehty tätä moottoria hyödyntäen. Pelimoottori tunnetaan myös graafisesti kauniista peleistä, kuten Kingdom Come: Deliverance ja Everybody's Gone to the Rapture. CryEnginestä on tällä hetkellä olemassa viisi versiota, joista viimeisin on CryEngine V. Vuonna 2016 CryEnginen ostanut Amazon julkaisi uuden monipuolisemman version CryEnginestä nimeltä Amazon Lumberyard. CryEngine V:llä pystyy kehittämään pelejä viidelle eri pelialustalle sekä Oculus Rift VR -alustalle. Pelimoottori on kehitetty pääosin 3D pelejä varten, mutta myös 2D-pelien tekeminen on mahdollista. Ohjelmointi CryEnginessä tapahtuu C++- ja Lua-ohjelmointikielillä. CryTekillä on oma kauppapaikka CryEnginen käyttäjille, CRYENGINE Marketplace, josta löytyy ostettavana tai ilmaiseksi muiden käyttäjien tekemiä resursseja. (Wikipedia: CryEngine 2018).

3.5.2 CryEngine ja Flash

CryEngine hyödyntää ulkoista väliohjelmistoa, nimeltä Scaleform GFx. Scaleformilla voi renderöidä ja pyörittää erikseen luotuja flash-objekteja. Sen avulla CryEnginessä voi käyttää Flash-objekteja käyttöliittymässä. (CryEngine User Interface 2014).

Flash-objektin luomiseen tarvitaan Adobe Flash -ohjelma tai jokin samantyyppinen ohjelma. Objektiin on mahdollista liittää esimerkiksi erilaisia funktioita, animaatioita ja tapahtumia. Flash-objektin luonnin jälkeen se tulee ensin paljastaa CryEnginelle xml-tiedoston avulla. Tämän jälkeen sitä ja siihen kiinnitettyjä ominaisuuksia pystyy käyttämään CryEnginen UI toimintojen ja visuaalisen skriptaustyökalun, Flowgraphin kautta. (CryEngine UI Element 2016).

3.5.3 CryEnginen käyttöliittymän toiminnallisuuksia

CryEnginessä on useita eri komponentteja UI:n tekemiseen. UI elementillä tarkoitetaa Flashillä tehtyjä Flash-objekteja, eli nimenomaan grafiikkaa ja animaatioita. UI toiminto on normaali CryEngine toiminnosta eroava komponentti, jota

käytetään käyttöliittymän luomisessa. Sitä käytetään UI flowgraphissa UI-animaatioiden ja -tapahtumien säätelemiseen. CryEnginessä on myös UI Event System, joka toimii flowgraphin ja itse pelikoodin kommunikointivälineenä. (CryEngine User Interface 2014).

3.6 Vertailua

3.6.1 Valmiit käyttöliittymäelementit

Monissa pelimoottoreissa, joissa voi luoda graafista käyttöliittymää, on valmiina käyttäjälle käyttöliittymäelementtejä. Näitä elementtejä voivat olla esimerkiksi valmiit painikkeet ja tekstikentät. Kaikista yllä mainituista pelimoottoreista Unreal Enginellä on laajin elementtitarjonta. Elementit on jaettu selkeästi eri kategorioihin, ja niihin on helppo päästä käsiksi. Joissain pelimoottoreissa, kuten Unityssa ja Adventure Game Studiossa, on valmiina muun muassa peruselementit, kuten painike, kuva, tekstikenttä ja liukusäädin. Näiden lisäksi Unitysta löytyy vierityspalkki esimerkiksi tekstikentälle ja UI Mask, jolla voi piilottaa osan UI-elementistä. Adventure Game Studiossa taas löytyy peruselementtien lisäksi pudotusvalikko, kustomoidut teksti-ikkunat ja kustomoitu inventaario. Koska Adventure Game Studio on tarkoitettu pääosin seikkailupelien tekemiseen, sen yksi oletetuin pelielementti on inventaario. Tämän takia pelimoottorin kehittäjät ovat helpottaneet käyttäjien työtä luomalla valmiin oletusinventaarion. Inventaariota voi tietysti muokata, tai sitten käyttäjät voivat luoda täysin uuden oman kustomoidun inventaarionsa. Tämä on etu verrattuna muihin pelimoottoreihin.

GameMaker Studiossa käyttöliittymän toteuttaminen toimii vähemmän visuaalisesti. Käyttöliittymän tekeminen tapahtuu joko raahaa ja pudota -ohjelmoinnilla tai kutsumalla funktioita koodissa. GameMaker Studion eri käyttöliittymäkomentoja ovat draw_sprite, draw_background, draw_text, draw_scaled_text, draw_rectangle, horizontal-/vertical_gradient, draw_ellipse, gradient_ellipse, draw_line ja draw_arrow. Näiden lisäksi eri toimintoihin, kuten tekstin tai taustojen piirtämiseen, on olemassa tarkempia ja muuten erilaisia funktioita. Vielä edistyneempiä funktioita ovat draw_circlen tuottamien ympyröiden kulmien

lisäämiseen tarkoitettu draw_set_circle_precision, yksinkertaisen painikkeen luova draw_button, yksinkertaisen health barin luova draw_healthbar sekä draw_path, joka renderöi polun pelimaailmaan paikasta A paikkaan B. Verrattuna muihin pelimoottoreihin GameMaker Studio antaa käyttäjälle monipuolisemman valikoiman piirtotoimintoja ja elementtejä. GameMaker Studio tarjoaa muun muassa valmiit liukuvärjäysominaisuudet, jotka toimivat esimerkiksi pelaajahahmon jäljellä olevan elämän esittämisessä.

CryEnginestä löytyy valmiita käyttöliittymäobjekteja käyttäjän käyttöön. Näitä ovat mm. painike, valintaruutu, pudotusvalikko, paneeli, vierityspalkki ja tekstikenttä. Objektien grafiikan ja toimintojen muokkaaminen tapahtuu Adobe Flash -ohjelmalla tai jollain samantyyppisellä ohjelmalla. Näin ollen käyttäjällä on monipuoliset mahdollisuudet muokata objekteja mieluisaksi.

3.6.2 Fontin muokattavuus

Yksi tapa kustomoida käyttöliittymää on vaihtaa sen tekstin fonttia. Unityn oletusfonttina on dynaaminen Arial, mutta on myös mahdollista muokata tekstiä muuttamalla se johonkin muuhun fonttiin tai tuomalla oman kustomoidun fontin. Tuotaessa omaa fonttia käyttäjä pääsee käsiksi fontin tuontiasetuksiin. Unity tukee TrueType- ja OpenType-fonttiformaattia. Tuontiasetuksista löytyy mm. fontin oletuskoko ja se, mitä fonttia käytetään varalla, eli jos kyseistä fonttia ei voi käyttää. Unity-sovelluksen käynnistyessä Unity generoi oletuksena fontista tekstuurin, jota se käyttää sovelluksen aikana. Sen sijaan, jos fontin tuontiasetuksissa määritellään fontti dynaamiseksi, Unity ei generoi alussa tekstuuria, vaan se generoi tekstuuria tarpeen mukaan sovelluksen ollessa käynnissä. Tämä vähentää muistin käyttöä, sillä kaikkia fontissa määriteltyjä merkkejä ei välttämättä käytetä sovelluksessa. Jos Unity ei löydä haluttua fonttia, se käy läpi vientiasetuksissa määritetyt varafontit. Jos näitäkään ei voida käyttää, Unity palaa sen omaan oletusfonttiin Arialiin, ja jos tätäkään ei löydy, se käyttää Unityyn itseensä ohjelmoitua Liberation Sans -fonttia. Liberation Sans on samannäköinen kuin Arial mutta yksinkertaisempi. Fontin väriä ja materiaalia on mahdollista muuttaa esimerkiksi Unityn materiaalien tai GUISkinin kautta.

Unreal Enginellä on samantyyppisiä ominaisuuksia. Unreal Engine tukee myös TrueType - sekä OpenType -formaatin fontteja. Yksi merkittävä huomio Unreal Enginen fonttikäytössä on font face assettien käyttö. Tämä tarkoittaa sitä, että fonttia tuotaessa Unreal Engine tallentaa sen tiedot font face assettiin, jotta sitä voidaan myöhemmin käyttää uudelleen useissa font asseteissa, mikä vähentää muistin käyttöä. Sekä Unreal Engine, että Unity ovat löytäneet tapoja vähentää muistin käyttöä fonttien suhteen. Unreal Engine tarjoaa Unityn tapaan fontin tyylittelyvaihtoehtoja, kuten fontin materiaalin tai tekstuurin vaihtoa, tekstin ääriviivojen lisäämistä sekä monipuolista värin vaihtoa.

GameMaker Studio on hieman erilainen kuin edellä mainitut. Pelissä käytetään aina oletuksena Arial-fonttia, ja fonttikokona on 12. Jos käyttäjä haluaa käyttää muuta fonttia, hänen tulee luoda uusi fonttiresurssi, jossa määritellään fontin asetukset, kuten nimi, käytettävä fontti, fontin koko, se, onko fontti lihavoitu tai kursivoitu, tekstuuriryhmä sekä halutun merkkijonon pituus (englanniksi “font range”). Toisin kuin edellä mainituissa pelimoottoreissa, joissa fontin tyylivalinnat ovat enemmän objektikohtaisia, GameMaker Studiossa tyylivalinnat ovat fonttikohtaisia. Valinnat, kuten lihavointi, määritellään itse fontin asetuksissa. Käytettäessä fonttia se tulee valita halutun GUI-objektin toiminnoissa. Tämän voi tehdä joko GameMaker Studion raahaa ja pudota -toiminnolla objektin asetuksissa tai koodin kautta käyttämällä funktiota draw_set_font(font). Fonttia voi vaihtaa milloin haluaa. GameMaker Studioon on myös mahdollista luoda täysin oma fontti lisäämällä spriten fontista projektiin.

Adventure Game Studiolla on kolme valmista oletusfonttia, joita käytetään eri objekteissa ja toiminnoissa. Font 0, eli ensimmäinen oletusfontti, on tarkoitettu viestilaatikoiden tekstiä varten. Font 1 on hahmojen puhetta varten olevan tekstin fontti, ja Font 2 on samanlainen kuin Font 1 mutta ääriviivoilla. Kun puheteksti niin sanotusti leijuu ilmassa hahmojen yläpuolella, on tärkeää, ettei se sulaudu taustaan. Tässä tapauksessa ääriviivallinen teksti on erittäin hyödyllinen. Ääriviivat on mahdollista saada aikaan kahdella tavalla: automaattisesti Adventure Game Studion omalla työkalulla tai tuomalla itse fontti, jossa on valmiiksi ääriviivat. Jos käyttäjä haluaa tuoda oman fonttinsa, sen tulee korvata yksi kolmesta oletusfontista tai sen asetuksista pitää määritellä, milloin sitä käytetään. Adventure Game Studio tukee TrueType- ja SCI-fonttiformaatteja. Näiden lisäksi se itse käyttää oletusfonteissaan WFN -formaattia.

CryEnginessä luodaan fonttikirjasto, joka sisältää kaikki pelissä käytettävät fontit. CryEngine tukee TrueType -fontteja. Kirjastoa luodessa voi päättää jokaisen fontin kohdalla, mitä ominaisuuksia haluaa pitää näistä. Esimerkiksi halutaanko kaikki fontin erityismerkit mukaan. Tämä vähentää pelin muistin käyttöä. Fonttikirjasto tulee olla Flash-objekti, joka sitten käännetään Scaleform GFx -formaattiin. Tämän jälkeen halutut fontit lisätään fonttikirjastosta käyttöliittymäobjekteihin.

3.6.3 2D- ja 3D-käyttöliittymän toteuttaminen

Pelissä voi olla joko 2D- tai 3D-käyttöliittymä. 2D-käyttöliittymä renderöidään kameralle tasaisena kuvana. 3D-käyttöliittymä sen sijaan renderöidään yleensä pelimaailmaan, ja sitä voi muokata esimerkiksi sijainnin ja rotaation avulla epätasaisemmaksi ja vaikka etäisemmäksi, mikä voi olla käyttäjän silmälle mieluisaa. Mainituista pelimoottoreista Unity, Unreal Engine ja CryEngine mahdollistavat 2D-käyttöliittymän käytön lisäksi myös 3D-käyttöliittymän. Tämä on myös mahdollista GameMaker Studiossa mutta hieman monimutkaisemmin. Adventure Game Studiossa tämä ei ole mahdollista ollenkaan.

3.7 Kunniamainintani

Tässä luvussa käsitellään vielä kahta muuta kahta pelimoottoria, jotka halusin ottaa mukaan käsittelyyn. Näillä moottoreilla kuitenkin tehdään enimmikseen tietynlaisia pelejä, eivätkä ne anna kehittäjälle täyttä vapautta. Tämän takia siirsin ne omaan kappaleeseensa. Kummallakin on kuitenkin mahdollista toteuttaa seikkailupelejä.

3.7.1 Ren’Py

Ren’Py Visual Novel Engine on ilmainen pelimoottori, jolla toteutetaan visual novel -pelejä. Visual novel on interaktiivinen tarina, ja se kuuluu seikkailupelien genreen. Visual novel on seikkailupelin alalaji, mutta siinä ei ole monipuolista

pelattavuutta, vaan pelin pelaamista voi verrata sellaisen kirjan lukemiseen, jossa on sekä tekstiä että kuva. Visual novel -peleillä voi olla useita eri loppuja, ja pelaaja voi tehdä päätöksiä pelin aikana, jotka vaikuttavat pelin lopputulokseen. Pelattavuuden vähäisyyden takia monet väittelevät siitä, voiko visual novel -peliä kutsua itseasiassa peliksi, vai onko se enemmänkin vuorovaikutettavaa fiktiota.

Ren'Pyn ensimmäinen versio julkaistiin vuonna 2004, ja sen kehitti Tom "PyTom" Rothamel. Tällä hetkellä uusin versio on v6.9.9.14.3, ja se julkaistiin 5. huhtikuuta 2018. Ren'Pyllä voi tuottaa pelejä kuudelle eri alustalle, ja sen skriptauskielenä toimii sen oma, ohjelmoinnin aloittelijoillekin helposti omaksuttava Ren'Py Scripting Language. Ren'Py tarjoaa mahdollisuuden muun muassa pelin tallennukseen, videoiden ja kuvien esittämiseen sekä pelin UI-elementtien visuaalisen ulkonäön ja animaatioiden muokkaamiseen. Muun muassa Indiegames.com, MakeUseOf ja The Guardian ovat suositelleet Ren'Py-pelimoottoria pelikehitykseen. (Wikipedia: Ren'Py 2018).

Kaiken kaikkiaan Ren'Py on hyvä työkaluvalinta, jos käyttäjä haluaa tehdä visual novel -pelin. Ren'Py on monipuolinen käyttöliittymämuokkauksessaan, ja sen skriptauskieli on helposti opeteltavissa ohjelmoinnin aloittelijalle. Isona etuna on myös se, että ohjelma on ilmainen. Ren'Py on kuitenkin vain visual novel -pelejä varten. Jos käyttäjä haluaa tehdä eri genren pelin, tämän täytyy etsiä toinen työkalu.

3.7.2 RPG Maker

RPG Maker -pelimoottorilla ei ole tekemistä aikaisemmin mainittujen GameMaker Studion ja Adventure Game Studion kanssa nimien samankaltaisuudesta huolimatta. Sen sijaan nimensä mukaisesti RPG Makerilla kehitetään tietokoneroolipelejä.

RPG Makerin ensimmäinen virallinen versio julkaistiin vuonna 1992 Japanissa nimellä RPG Tsukūru Dante 98. Tämä ei kuitenkaan ollut RPG Tsukūru Dante 98:n kehittäjäyrityksen ASCIIn ensimmäinen RPG Makerin versio, vaan sen ensimmäisiä versioita kehitettiin jo vuonna 1988. RPG Tsukūru Dante 98:n jälkeen on julkaistu seitsemän tietokoneversiota. Vuonna 2002 osa ASCIIsta muodosti

oman yrityksen ja julkaisi RPG Maker 2003:n. Pelimoottori on myös julkaistu erilaisille konsoleille, kuten PlayStation 2 ja Nintendo 3DS. Tämänhetkinen uusin tietokoneversio on RPG Maker MV, jonka julkaisi Degica vuonna 2015. (Wikipedia: RPG Maker 2018).

RPG Maker MV:n voi ladata Windows-, Mac- ja Linux (Steam, Ubuntu) -käyttöjärjestelmille, ja sillä voi kehittää pelejä viidelle eri alustalle. Skriptaus tapahtuu JavaScript- ja Html5-kielillä. RPG Maker MV tarjoaa sekä tietokoneen hiirituen että kosketusnäyttötuen, vuoropohjaisen taistelujärjestelmän, mahdollisuuden 2000 eri peliesineen luomiseen ja käyttöön, käyttöliittymäelementtien grafiikan vaihdon, jne. (RPG Maker 2018).

RPG Makerilla on kattavat ohjeet ohjelman käyttöön sekä aktiivinen foorumi täynnä ihmisiä, joilta voi kysyä apua. Pelimoottorilla tehdään pääosin kuitenkin vain tietynlaisia RPG-pelejä. Kaikki RPG Maker -versiot ovat olleet melko kalliita. Pelimoottori itsessään on melko voimakas ja sen kehittänyt yritys on vaikutusvaltainen, joten hinta ei ole yllätys. RPG Makerin moton ”Simple enough for a child. Powerful enough for a developer.” (RPG Maker 2018) mukaan moottori on tarpeeksi voimakas kehittäjälle mutta myös tarpeeksi yksinkertainen, jotta lapsikin voi sitä käyttää. RPG Maker on hyvä aloitus pelinkehityksen aloittelijoille, kunhan pystyy sijoittamaan rahaa ohjelman ostoon.

4 Käytännön toteutus

4.1 Suunnittelu

Käytännön toteutus on lyhyt esimerkki point and click -tyylisestä seikkailupelistä ja sen käyttöliittymästä. Peli on toteutettu Unitylla, ja siinä on käytetty Unity Asset Storesta ostettua SchrippleA:n kehittämää Point and Click Adventure Game Pack -pakettia.

Point and click -seikkailupelit ovat yksi tunnetuimmista seikkailupelien alalajeista. Point and click tarkoittaa suomeksi käännettynä ”osoita ja klikkaa”, mikä kertoo pelin päätoiminnallisuudesta. Näitä pelejä ohjataan yleensä hiirellä, jolla valitaan ja klikataan eri objekteja, kuten hahmoja tai esineitä. Klikkauksen seurauksena

on yleensä jokin tapahtuma. Esimerkiksi jos pelaaja klikkaa hahmoa, tämä voi aloittaa dialogin. Kun pelaaja klikkaa tiettyä objektia, siitä seuraava tapahtuma voi myös viedä tarinaa eteenpäin. Point and click -peliin voi kuulua myös esineiden käyttö, jota käsitellään alaluvussa 2.5.5. Esine valitaan pelaajan inventaariosta, jolloin se on aktiivisena esineenä, ja kun pelaaja seuraavan kerran klikkaa objektia, jonka kanssa voi vuorovaikuttaa, peli reagoi siihen.

Päätin valita käytännön osuuteeni point and click -pelin, koska ne ovat hyvin tuttuja minulle ja myös melko yksinkertaisia toteuttaa. Se tuntui hyvältä vaihtoehdolta myös siksi, että kuten jo mainitsin, point and click -pelit ovat yksi tunnetuimmista seikkailupelien alalajeista. Neljäs peruste oli se, että point and click -peleissä grafiikka on yleensä kaksiulotteista. Pystyn itse tuottamaan 2D-grafiikkaa, ja se on minulle tutumpaa kuin kolmiulotteiset mallit.

Pelin suunnittelu lähti työkalun päättämisestä. Koulun kautta minulla on eniten kokemusta Unitysta, joten se teki siitä hyvän vaihtoehdon. Tunsin myös, että Unity voisi olla tarkoitukseeni paras, sillä pelin päätarkoituksena on käyttöliittymän esittely, ja Unityn käyttöliittymän muokkaustoiminnot ovat monipuoliset. Osasin käyttää niitä jo valmiiksi, ja niillä saa painikkeista ja muista käyttöliittymäelementeistä juuri sellaiset kuin haluaa. Unity oli myös hyvä vaihtoehto sen vuoksi, että toisin kuin esimerkiksi GameMaker Studiossa, Unityn käyttöliittymän muokkaus on enemmän visuaalista, mikä on minulle mieluisampaa. Nämä yhdessä tekivät Unitysta omaan tilanteeseeni parhaimman vaihtoehdon. Unity myös tarjoaa vaihtoehdon toteuttaa kolmiulotteisen käyttöliittymän, jos jatkossa muuttaisin mieleni ja kokeilisin toteuttaa hieman erilaista point and click -peliä. Aloitin käytännön toteutuksen Unity 5.6.0 -versiolla, mutta päivitin lopulta Unity 2017.4.3 -versioon.

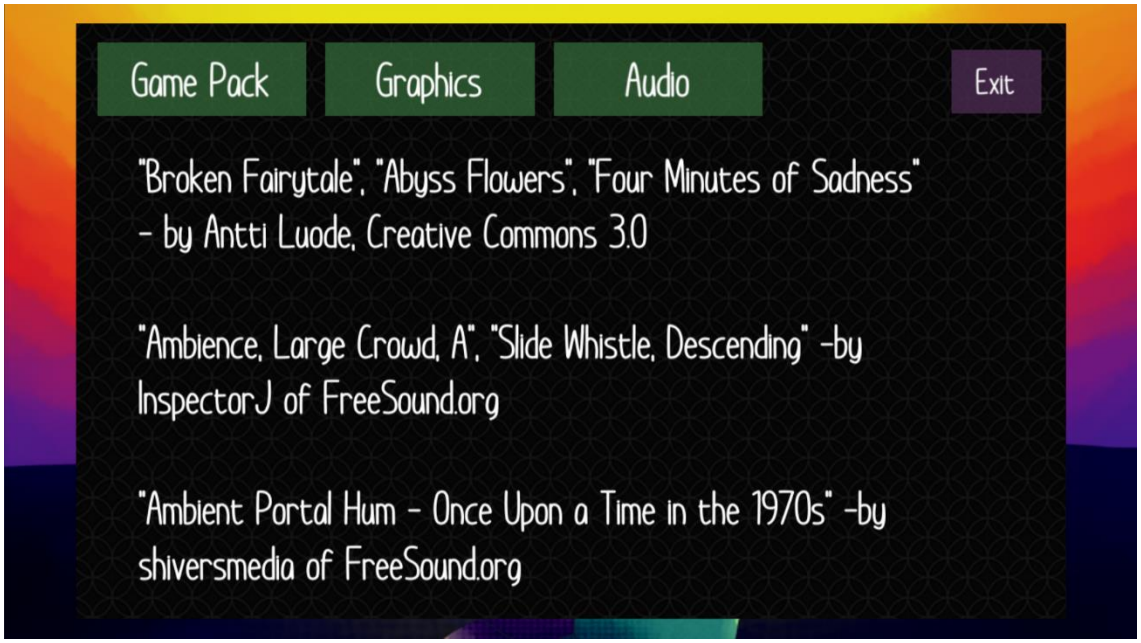
Vaikka pelit ovat kiinnostukseni kohde, pelisuunnittelu ei ole vahvin taitoni. Tämän takia minulla oli hankaluuksia päättää, mitä pelissä tapahtuu ja miltä pelin käyttöliittymä näyttää visuaalisesti. Toteutan pelissä muutaman käyttöliittymäelementin, joita listasin alaluvussa 2.5.5. Elementit on valittu sen mukaan, mikä sopii kyseiseen pelityyppiin. Point and click -pelissä esimerkiksi kompassi ei välttämättä ole kovin hyödyllinen, sillä pelikentät ovat yleensä vain kaksiulotteisia. Kompassin voi kuitenkin integroida point and click -peliin esimerkiksi sellaiseen tilanteeseen, että pelaaja on keskellä risteystä ja hänelle on mahdollista kääntyä

katsomaan joka suuntaan. Tällöin kompassi voi olla näyttämässä, mihin ilmansuuntaan ollaan katsomassa. Elementit, jotka otin peliini mukaan, ovat kartta, tavoitteen näyttäminen ja inventaario. Näiden lisäksi käyttöliittymässä on mukana ajan näyttäminen, päävalikko ja loppuvalikko. Osa elementeistä, kuten kartta, eivät ole kovin hyödyllisiä tämänhetkisessä demossa, mutta minulla on ideoita siitä, kuinka sitä voi hyödyntää jatkokehityksessä. Pelissä voisi olla esimerkiksi paljon monimutkaisempi alue, jossa on useita risteyksiä, ja kartalla voi myös näyttää tiettyjen hahmojen olinpaikat, varsinkin jos ne vaihtavat sijaintia. Kartan kautta näkee myös sen hetkisen tavoitteen (kuva 9). En ottanut peliin mukaan esineiden manuaalista käyttöä, vaan jos pelaajalla on esine inventaariossa, muut hahmot reagoivat siihen automaattisesti. Syy tähän on se, että kyseistä toiminnallisuutta ei ollut käyttämässäni point and click -työkalussa, enkä halunnut ottaa riskiä toiminnallisuuden kehittämisessä, sillä pelissä oli jo muutenkin tarpeeksi haastetta.

Kuva 9. Kartta ja pelaajan tavoite. (Kuva: Vera Löfberg)

Olin päättänyt toteuttaa osan grafiikasta itse, mutta musiikkipuolesta minulla ei ollut paljoa tietoa. Satuin kuitenkin löytämään Antti Luoden Google Drive -linkin, jossa oli yli 1 000 vapaassa käytössä olevaa musiikkikappaletta. Tämän lisäksi tiesin, että vapaasti käytettävää musiikkia löytyy incompetech.com-sivustolta, jonne säveltäjä Kevin McLeod on säveltänyt ja ladannut noin 2 000

musiikkikappaletta. Grafiikoista taustavalokuvat ovat CC0-lisenssin alla julkaistuja kuvia tai muita kuvia, joita saa muokata ja käyttää, kunhan nimeää alkuperäisen tekijän tuotteessa. Krediitit ovat esillä pelissä (kuva 10). Piirretyt grafiikat ovat omaa työtäni.



Kuva 10. Kuva käytännön toteutuksen krediittivalikosta. (Kuva: Vera Löfberg)

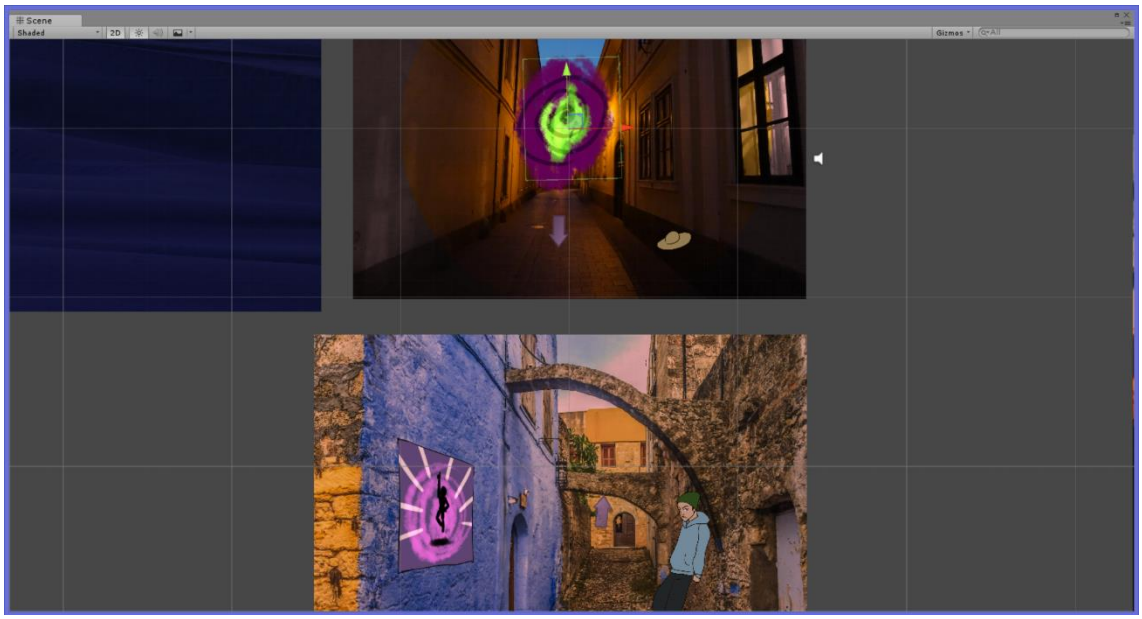
Halusin, että pelilläni olisi jokin tarkoitus. Päätin esittää pelaajalle moraalisen dilemman varastamisesta. Pelissä pelaaja tarvitsee tietyn esineen, jotta hän voi toteuttaa pelin päätarkoituksen. Esine on erään naisen omistuksessa, ja tämä taas on kadottanut sormuksensa. Pelaajalla on nyt edessään valinta: varastaako hän esineen naisen huomaamatta vai etsiikö hän naisen sormuksen. Sormuksen etsiminen tarkoittaa enemmän työtä, mutta se on rehellistä. Esineen varastaminen taas on helppoa ja nopeaa. Moraaliset dilemmat peleissä ovat aina kiehtoneet minua, sillä ne saavat ihmiset ajattelemaan asiaa tai dilemma voi jopa vaikuttaa heihin ihmisinä. Tällä hetkellä pelistä on tehty ensimmäinen taso, mutta haluan jatkokehittää peliä lisäämällä siihen useamman erilaisen tason ja syvemmän juonen. Inspiraation lähteinä toimivat point and click -pelit kuten Primordia ja Deponia (kuva 11).

Kuva 11. Kuva Deponiasta. Kursori muuttaa tekstuuria, kun se on vuorovaikutettavan objektin päällä ja sen vieressä oleva teksti kertoo, mikä objekti on. (Kuva: Vera Löfberg)

4.2 Kehitysprosessi

Point and Click Adventure Pack on paketti työkaluja, jotka mahdollistavat point and click -tyylisen pelin kehittämisen. Paketin kursoritekstuurin vaihdokseen tarkoitettu työkalu mahdollistaa tekstuurin vaihdon riippuen siitä, minkä kohdalla kursori on. Esimerkiksi, jos kursori on vuorovaikutettavan hahmon kohdalla, se vaihtaa tekstuuriin, joka kertoo pelaajalle, että tämä voi vuorovaikuttaa sen hahmon kanssa. Vuorovaikutuksen mahdollisuudesta kertoo myös toinen työkalu, joka mahdollistaa kohteen nimen esittämisen kursorin vieressä. Näiden lisäksi paketin dialogityökalun avulla voi helposti lisätä eri kohteille dialogilaatikon ja sen sisältämän dialogitekstin. Paketti sisältää myös oman tavan vaihtaa pelinäkymää. Jokainen haluttu näkymäpeliobjekti lisätään listaan, josta kentänvaihtoskripti pystyy tarkistamaan, minne ollaan siirtymässä (kuva 12). Tämän työkalun takia pelissä ei ole tarvetta käyttää useampaa Unity-skeneä. Paketissa on myös valmiudet käyttää liikutettavaa pelaajahahmoa. Näiden toimintojen lisäksi paketissa on valmis demoprojekti, sen assetit valmiina käytettäväksi ja ohjeet työkalujen käyttöön. Käytännön toteutuksessa käytin paketin työkaluista objektien nimien

näyttämistä, dialogijärjestelmää (kuva 13) ja näkymänvaihtoa. Jokaista skriptiä on muutettu melko paljon, jotta ne soveltuisivat käytännön toteutuksen peliin.

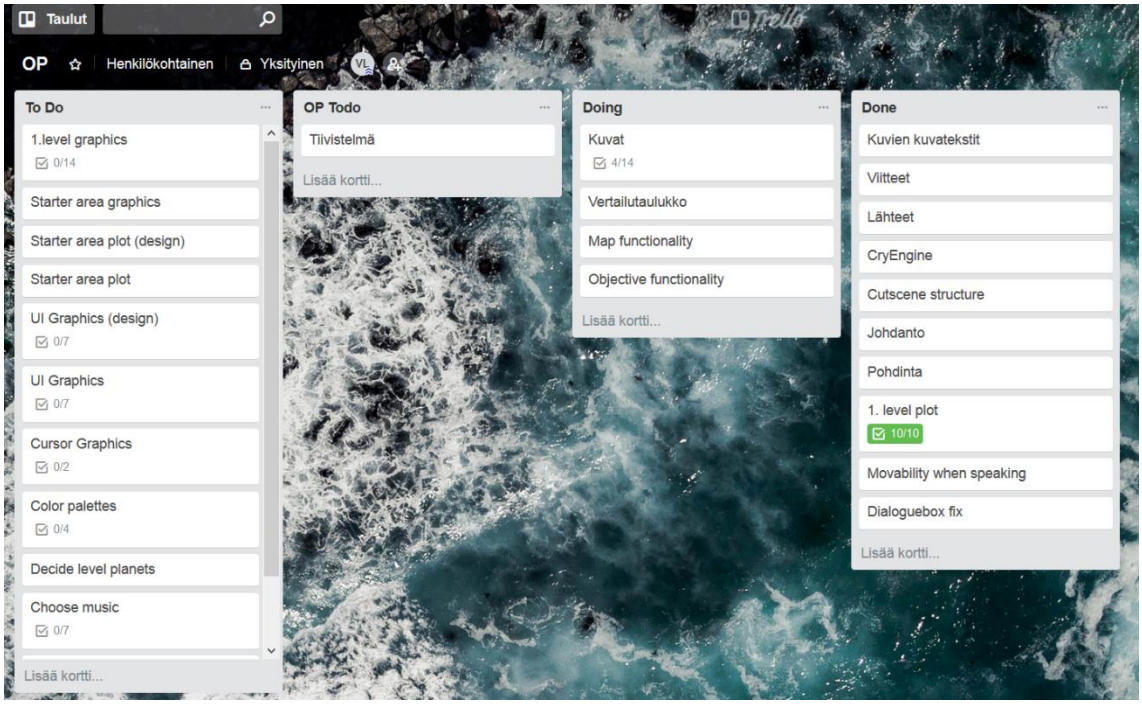


Kuva 12. Kuva Unityn editorista. Käyttöliittymässä näkyviä nuolia käytetään eri näkymien välillä liikkumiseen. (Kuva: Vera Löfberg)

Kuva 13. Esimerkki dialogijärjestelmästä ja noitahahmon nimen näyttämisestä. (Kuva: Vera Löfberg)

Pelin kehityksen alussa kohtasin hieman hankaluuksia, sillä paketin työkalut eivät olleetkaan täysin sellaisia, joita olisin toivonut; olisin halunnut peliin ohjailtavan pelihahmon, mutta luovuin sitten tästä ideasta. Hahmon liikuttaminen, ja tarkemmin sanottuna niiden alueiden rajaaminen, joille pelihahmo ei saanut mennä, ei onnistunut. Työkaluihin tutustuessani pääsin kuitenkin suurimpien ongelmien yli. Olen ollut myös yhteydessä paketin kehittäjään, SchrippleA:han, joka oli valmis auttamaan ongelmatilanteissa.

Projektin edetessä otin projektinhallintatyökaluksi Trellon (kuva 14), jota olin käyttänyt aikaisemmissakin projekteissani. Projektin yksinkertaisuuden takia päättelin, että Trello on riittävä työkalu käyttööni projektin ajaksi. Kun olin saanut kehitettyä tarpeeksi hyvän perustan pelille, kopioin sen myös GitHubiin ja päivitin sen aina, kun olin tehnyt jotain uutta. GitHubin otin käyttöön siksi, että halusin paikan, johon voin varmuuskopioida projektin ja josta voin tarvittaessa palauttaa projektin vanhan version, jos jotain menee pieleen. GitHub oli myös tuttu koulun kautta, joten se tuntui luontevalta päätökseltä.



Kuva 14. Käytin Trelloa siihen, että tietäisin mitä olen tehnyt ja mitä teen seuraavaksi. (Kuva: Vera Löfberg)

Koska peli alkaa välianimaatiolla ja itsestään aktivoituvalla dialogilla, päättelin että voisi olla järkevää jakaa aloitus ja itse peli eri osiin. Näillä osilla on muun muassa omat dialogiskriptit ja kamerat. Tällä tavalla sain vältettyä vielä monimutkaisemmat skriptit. Aloitus ja peli ovat samassa skenessä. Koska peli ei ole kovin iso, se että käyttää vain yhtä skeneä helpottaa esimerkiksi pelin uudelleenkäynnistämistä. Välianimaation keskikohdalla kamera siirtyy päävalikkoon, josta pelaaja voi valita painikkeiden avulla pelin aloittamisen, krediittien näyttämisen tai pelin lopettamisen. Krediittipainike avaa objektin, joka esittää krediitit näytöllä.

Point and click -peleissä on olennaista, että pelaaja tietää, minkä esineen tai hahmon kanssa voi vuorovaikuttaa. Omassa pelissäni käytettävät taustat ovat valokuvia, kun taas hahmot ja esineet, joiden kanssa voi vuorovaikuttaa, ovat piirrettyä grafiikkaa. Tämä valinta johtui siitä syystä, että en löytänyt hyvää taustagrafiikkaa käytettäväksi enkä itse ole hyvä tuottamaan taustoja. Tälläinen erottelu oli kuitenkin hyödyllinen ratkaisu, sillä näin pelaaja pystyy helposti havaitsemaan, minkä kanssa voi vuorovaikuttaa.

Peliä kontrolloi pääosin GameController-objekti, johon on kiinnitetty GameControllerScript ja AudioScript, ja ensimmäisen välianimaation suorittavat FirstCutSceneScript ja CutSceneDialog. Tarinankulku perustuu siihen, mikä pelitila on voimassa kullakin hetkellä. Pelitila on enum-tyyppinen muuttuja, johon on määritelty kaikki pelin tärkeimmät tapahtumat. Siten, dialogia käydessä hahmojen skriptit tarkistavat, mikä tila on käynnissä, ja näin tietävät mitä sanoa.

Paketin dialogijärjestelmä oli melko yksinkertainen, ja sen vuoksi kahden hahmon vuorottainen keskustelu ei ole mahdollista ilman puhujan ilmaisemista tekstissä. En tuntenut tämän olevan suuri ongelma, joten jätin pelaajahahmon ikään kuin mykäksi. Tämä tarkoittaa sitä, että pelaajahahmon puhe ei näy tekstinä, mutta muut hahmot reagoivat ikään kuin pelaajahahmo puhuisi. Dialogiskriptiä muokatessa huomasin myös toisen ongelman. Osa hahmoista toistaa aikaisempien pelitilojen dialogit ennen kuin sanovat, mitä sen hetkisessä tilassa tulisi sanoa. En ole löytänyt tähän vielä ratkaisua, mutta en tunne, että se on peliä rikkova ongelma. Kolmas pieni ongelma on se, että kursori liikkuu hieman itsekseen silloin tällöin. Tähänkään en ole löytänyt vielä ratkaisua, mutta jatkan näiden ongelmien selvittämistä.

Käytännön toteutusta tehdessäni päätin myös käyttää muutamaa animaatiota. En ollut näitä käyttänyt ennen Unityssa, joten alku oli hieman opettelua. Unityn animaatiotyökalu on kuitenkin melko yksinkertainen, joten opettelu oli melko mukavaa. Eri animaatioita on käytetty alkuosan välianimaatiossa, portaaleissa ja ”Shady Lady” -hahmon animoinnissa.

En halunnut käyttää Unityn oletusfonttia Arialia sen tavanomaisuuden vuoksi. Internet on täynnä vapaasti käytettäviä fontteja, ja hetken etsimisen jälkeen löysin fontin nimeltä Im Wunderland. Fontti oli uniikki mutta silti tarpeeksi helppolukuinen (kuva 15). Käytin fontin lihavoitua versiota, sillä normaaliversio oli hieman liian ohutta.

Kuva 15. Fonttia käytetään sekä dialogitekstissä, että käyttöliittymäpainikkeissa. (Kuva: Vera Löfberg)

4.3 Lopputulos

Käytännön toteutuksen päämäärä oli toteuttaa demo point and click -tyylisestä pelistä, joka esittelee sekä sen, minkälainen point and click -seikkailupeli voi olla, että sen, millainen käyttöliittymä tällaisessa pelissä voisi olla. Onnistuin tavoitteissani, ja lopputuloksena on peli, jossa on yksi kenttä ja kaksi hieman erilaista loppua. Tällaisenaan en julkaise peliä missään, mutta jatkokehityksen myötä voi tulla sellainen tilanne, että pelin voi jakaa laajemminkin. Tällä hetkellä pelialusta

on Windows Standalone, mutta minulla olisi myös kiinnostusta muokata pelistä WebGL-versio. Muita jatkokehityksen kohteita käsittelen seuraavan luvun eli pohdinnan lopussa.

5 Pohdinta

Aihetta tutkiessani huomasin, että ihmiset voivat helposti aliarvioida käyttöliittymän tärkeyden. Käyttöliittymän visuaalinen ilme ja käytännöllisyys voivat vaikuttaa käyttäjän mielipiteeseen koko ohjelmasta tai pelistä ja voivat pahimmassa tapauksessa jopa saada käyttäjän lopettamaan kyseisen sovelluksen käytön.

Säännöt ja ohjeistukset siihen, millainen on hyvä käyttöliittymä, eivät ole radikaalisti muuttuneet vuosien varrella. Käsitys siitä, mikä näyttää hyvältä ja miten muodot hahmottuvat, muodostuu ihmisen aivoissa. Ja mitä enemmän me aivoista opimme, sitä paremmin pystymme ymmärtämään, miten saamme aikaan ihmiselle mieluisan käyttöliittymän.

User Interface for Programmers -kirjassa Spolsky (2001, xiii) kertoo siitä, kuinka suurin osa hänen tutuistaan, jotka ohjelmoivat C++-kielellä, vihaavat käyttöliittymän ohjelmointia. Hän myös mainitsee, että nämä ihmiset mahdollisesti jopa pelkäävät käyttöliittymän suunnittelua. Uskon itsekin tämän olevan totta sekä omasta että muiden tapaamieni ohjelmoijien kokemuksesta. Ohjelmoijat saattavat haluta pitäytyä pelkästään teknisessä puolessa, kuten itse toimintojen ohjelmoinnissa, kun taas käyttöliittymäsuunnittelu on graafikoiden ja muiden suunnittelijoiden tehtävä. Pelko käyttöliittymien suunnittelua kohtaan voi olla perusteltu. Jos esimerkiksi käyttäjä ohjelmaa käyttäessään painaa väärää painiketta käyttöliittymässä tai ei löydä tiettyä toiminnallisuutta käyttöliittymästä, se ei ole käyttäjän vika vaan käyttöliittymän suunnittelijan. Tämä lisää paineita käyttöliittymän suunnittelussa.

Tutkiessani hahmolakeja huomasin, että jokainen laki tuntuu itsestään selvältä ja täysin loogiselta. Silti itse käyttöliittymää tehdessä niitä ei aina huomaa tietoisesti pohtia. Käyttöliittymän suunnittelu on paljon helpompaa ja lopputulos parempi, jos hahmolait pitää mielessä prosessin alusta loppuun.

Pelimoottoreiden tutkinnan ja vertailun jälkeen tulin siihen lopputulokseen, että esittelemistäni pelimoottoreista Unityn yleiset ja käyttöliittymään liittyvät ominaisuudet tekevät siitä parhaimman vaihtoehdon seikkailupelin toteuttamiseen. Unityn käyttöliittymätoiminnot ovat hyvät, sillä muokkausvaihtoehtojen monipuolisuus, animointimahdollisuudet ja visuaalinen toteuttaminen mahdollistavat sen, että käyttäjä saa käyttöliittymäelementeistään juuri sellaiset kuin haluaa. Näiden lisäksi Unity on melko aloittelijaystävällinen ja antaa useita ohjeita käyttöliittymän ja pelien tekemiseen. Unityn dokumentaatiota päivitetään kuitenkin melko hitaasti, minkä takia osa ohjeista on vanhemmille versioille. Myös Unreal Enginessä on samat ominaisuudet, mutta verrattuna Unityyn Unreal Engine on hieman monimutkaisempi sen blueprienttien takia. Kuitenkin jos Unreal Enginen blueprintteja osaa käyttää, ne tarjoavat erittäin monipuolisen visuaalisen tavan toteuttaa eri toiminnallisuuksia. Unreal Enginessä on myös kattavat valmiit käyttöliittymäelementit käyttöliittymän helppoon toteutukseen.

Myös GameMaker Studiossa on sen oma helppo drag and drop -tyylinen visuaalinen tapa toteuttaa eri toiminnallisuuksia. GameMaker Studion muut ominaisuudet ovat kuitenkin vähemmän visuaalisia, mikä heikentää sen käyttäjäystävällisyyttä. Myös Adventure Game Studion käyttöliittymänmuokkaus ja muut toiminnallisuudet ovat vähemmän visuaalisia, ja esimerkiksi sen fonttien rajoitus vähentää pelimoottorin miellyttävyyttä. Näistä huolimatta Adventure Game Studio on kuitenkin hyvä vaihtoehto omaan yksinkertaisempaan seikkailupeliprojektiin.

CryEngine eroaa muista pelimoottoreista siinä, että sillä voi käyttää Adobe Flash -assetteja projektissa. Flash on laajasti käytetty teknologia, joten monella on siitä kokemusta. Tämän takia käyttöliittymäelementtien ja seikkailupelin toteuttaminen CryEnginellä on hyvä vaihtoehto Flashin taitajalle. Adobe Flash on kuitenkin vanhentunut, ja vuonna 2017 Adobe ilmoitti, että Flash Player on poistumassa käytöstä ja jaosta vuonna 2020. (Wikipedia: Adobe Flash) CryEnginen pohjalta toteutettu, vuonna 2015 julkaistu Amazon Lumberyard -pelimoottori on samankaltainen kuin CryEngine, mutta siitä on poistettu Flashia käyttävä Scaleform GFx. (Wikipedia: Amazon Lumberyard).

Sen, mikä on “paras” pelimoottori, määrittää kuitenkin ennen kaikkea käyttäjän tarve. Jos käyttäjä haluaa tehdä seikkailupelin työkalulla, jossa on mahdollisimman paljon vapautta, hänen kannattaa valita esimerkiksi Unity tai Unreal Engine.

Mutta jos käyttäjä haluaa tehdä esimerkiksi visual novel -pelin, hän hyötyy eniten Ren'Pystä. Työkaluja ja pelimoottoreita löytyy todella moneen käyttöön ja jokaisella niistä on kuitenkin omat positiiviset ja negatiiviset ominaisuutensa, joten jää käyttäjän päätökseksi, mistä hän itse pitää eniten.

Iso osa käyttöliittymäelementeistä käytännön toteutuksessa ovat ainakin jonkin verran läpinäkyviä. Seikkailupelissä, jossa käytetään grafiikoita visuaalisessa tarinankerronnassa, on tärkeää, että itse peli näkyy tarpeeksi käyttöliittymän alta. Käyttöliittymän osittaisella läpinäkyvyydellä mahdollistin sen, että pelaaja näkee suurimman osan ruutua suurimman osaa ajasta. Tämän ansiosta käyttöliittymä ei vie liikaa tilaa, vaikka se olisikin suuri.

Alaluvussa 2.5.4 esittelin diegeettistä teoriaa. Sen mukaisesti käytännön toteutuksen pelissä käytetään ei-diegeettistä käyttöliittymää eli käyttöliittymää, joka ei ole mukana tarinankerronnassa eikä pelimaailmassa.

Alaluvussa 2.5.1 käsittelin yleisiä videopelikäyttöliittymän sääntöjä. Kuten yllä mainitsin, käytännön toteutuksen käyttöliittymä on osittain läpinäkyvä, mikä vähentää sitä, että se veisi liikaa huomiota pelistä. Käyttöliittymä on selkeä ja yksinkertainen, ja käyttöliittymän hillitty ulkoasu ja värimaailma sopivat pelin tyyliin. Halusin ensin käyttää pelkästään mustaa väriä ja harmaan sävyjä, mutta päädyinkin sitten lisäämään joitain muita värejä, kuten tumman violetin, joka on myöskin hillitty mutta tuo enemmän persoonallisuutta peliin. Käyttöliittymää suunnitellessa pidin myös mielessä alaluvun 2.3 vältettävät asiat. Tein esimerkiksi päävalikon ja pelin käyttöliittymän navigoinnista mahdollisimman yksinkertaista ja helppoa pelaajalle. Kaikissa painikkeissa on tekstit, jotka kertovat, mitä tapahtuu, kun niitä painaa. Myös käyttöliittymän responsiivisuuteen on kiinnitetty huomiota: käyttöliittymäelementit skaalautuvat sen mukaan, minkä kokoinen peliruutu on. Tämä on mahdollistettu Unityn ankkureilla ja Canvas-objektin valmiilla CanvasScaler-skriptillä.

Sovelsin muutamaa hahmolakia käytännön toteutuksessa; hahmolakeja käsiteltiin alaluvussa 2.2. Alkunäkymän päävalikossa on käytetty samankaltaisuuden ja läheisyyden lakeja, sillä valikkopainikkeet ovat samanlaisia ja ne on ryhmitelty samaan paikkaan. Samaa ideaa on käytetty myös krediittivalikossa. Pelissä käytettyjen resurssien krediitit on jaettu kolmen painikkeen taakse, jotka ovat ”Game Pack”, ”Audio” ja ”Graphics”. Näiden painikkeiden kanssa on myös sovellettu

samankaltaisuuden ja läheisyyden lakeja. Ne ovat kuitenkin hieman erilaisen näköisiä verrattuna päävalikon painikkeisiin, jotta pelaaja saisi vielä selvemmin kuvan siitä, että painikkeet ovat omat ryhmänsä. Krediittivalikossa on myös käytetty kohde ja alusta -lakia, sillä kun krediittivalikko on näkyvissä, taustalla oleva valikkokuva tummennetaan. Pelin inventaariossa on myös käytetty läheisyyden ja samankaltaisuuden lakeja esinepaikkojen kohdalla. Tämä kertoo pelaajalle, että inventaario on yksi kokonaisuus, ja koska esinepaikat ovat samanlaiset, pelaaja tietää, että esineitä voi olla useampia. Myös inventaariota rajaavilla viivoilla on tarkoitus. Tässä on käytetty sulkeutuvuuden lakia, minkä ansiosta inventaarion kaksi viivaa näyttävät sitä ympäröivältä suorakulmiolta.

Halusin tehdä pelin päävalikosta melko yksinkertaisen. Valikosta löytyy pelin nimi ja painikkeet pelin aloitukselle, lopetukselle ja krediittien näyttämiselle. Jokaisessa painikkeessa on tekstit, joista voi helposti päätellä, mitä painikkeesta tapahtuu. Krediittipainikkeen takaa löytyy toinen valikko, jossa on listattu kaikki resurssit, jotka vaativat tekijän nimeämisen. Tässä valikossa olen käyttänyt samankaltaisia painikkeita kuin päävalikossa mutta eri värisinä ja eri kokoisina.

Jokaisella painikkeella on tietyt animaatiot. Kun kursori on painikkeen päällä, painikkeen väri muuttuu vaaleammaksi, ja kun painiketta klikataan, se muuttuu tummemmaksi. Tämä kertoo pelaajalle, että hän on vuorovaikuttamassa painikkeiden kanssa.

Onnistuin käyttöliittymän toteutuksessa mielestäni hyvin mutta en täydellisesti. Varsinkin käyttöliittymän värien valitseminen on hankalaa ja vaatii paljon pohdiskeluaikaa. Itse valitsin turvallisemman vaihtoehdon ja käytin hillittyjä värejä ja läpinäkyviä grafiikoita. Haluan kehittää tätä vielä tulevaisuudessa, kun omaksun paremmin eri värien käytön. Valokuvien käyttö taustana ei ollut ensimmäinen vaihtoehtoni, mutta pidän sen antamasta visuaalisesta ilmeestä. Uskon, että tästä sai hyvän ominaisuuden peliin. Jatkossa luultavasti etsin tai valokuvaan itse paremmat kuvat taustagrafiikoiksi. Pelinäytön alueen tehokas käyttäminen ja käyttöliittymäelementtien taitava sijoittelu ovat myös kullanarvoinen taito oppia, mutta se kehittyy varmasti kokemuksen myötä.

Yksi jatkokehityksen kohde on myös esineiden manuaalinen käyttö. Tällä hetkellä esineiden käyttö tapahtuu automaattisesti pelaajan vuorovaikuttaessa objektien

kanssa, mutta jatkossa haluan, että pelaajan tulee itse miettiä, mitä esinettä hän milloinkin käyttää.

Toinen jatkokehityksen kohde on esineiden manuaalinen käyttö. Tällä hetkellä esineiden käyttö tapahtuu automaattisesti pelaajan vuorovaikuttaessa objektien kanssa, mutta jatkossa haluan, että pelaajan tulee itse miettiä, mitä esinettä hän milloinkin käyttää.

Valokuvien käyttö taustana ei ollut ensimmäinen vaihtoehtoni, mutta pidän sen antamasta visuaalisesta ilmeestä. Uskon, että tästä saa hyvän ominaisuuden peliin. Jatkossa luultavasti etsin tai valokuvaan itse paremmat kuvat taustagrafiikoiksi.

Kuten jo aikaisemmin mainitsin, tämänhetkinen kartta ei ole kovin hyödyllinen. Jatkossa kuitenkin haluaisin hioa siitä paremman työkalun pelaajalle. Haluan tehdä monimutkaisemman pelin, ja se yleensä tarkoittaa myös isompia pelialueita. Isoilla pelialueilla voi olla vaikeaa muistaa missä mikin pelihahmo ja paikka on, minkä takia kartta on tärkeä väline pelaajalle.

Pelistä löytyy vielä joitain pieniä bugeja, joihin en itse ole löytänyt ratkaisua. Tulevaisuudessa voi kuitenkin olla, että muiden ihmisten ja internetin resurssien avulla pääsen näistäkin ongelmista yli.

6 Lähteet

Adventure Game Studio Manual. 2018. http://www.adventuregamestudio.co.uk/manual/
Other Features -> Editing the GUIs
Tutorial -> Setting up the Game -> Inventory

GameMaker Studio. 2018. The Draw GUI Event. https://docs.yoyogames.com/source/dadiospice/000_using%20gamemaker/events/draw%20gui%20event.html

GameMaker Studio. 2018. Drawing. https://docs.yoyogames.com/source/dadiospice/002_reference/drawing/index.html

GameMaker Studio. 2018. Drawing Basic Forms. https://docs.yoyogames.com/source/dadiospice/002_reference/drawing/drawing%20basic%20forms/index.html

Johnson, J. 2010. Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules. Morgan Kaufmann.

Laine, A. 2004. Hahmolait käytettävyyden parantajina. Jyväskylän yliopisto. http://www.mit.jyu.fi/opetus/opinnayte/LuK/Hahmolait/

Lecky-Thompson, G. 2008. Video Game Design Revealed. Course Technology.

Mitchell, B. L. 2012. Game Design Essentials. John Wiley & Sons, Incorporated.

RPG Maker. 2018. RPG Maker MV. http://www.rpgmakerweb.com/products/programs/rpg-maker-mv

Russell, D. 2011. Video game user interface design: Diegesis theory. Dev.Mag. http://devmag.org.za/2011/02/02/video-game-user-interface-design-diegesis-theory/

Sarah, B. 2017. Mockplus. https://www.mockplus.com/blog/post/bad-ui-design-examples

Siang, T. 2018. Interaction Design Foundation. https://www.interaction-design.org/literature/article/bad-design-vs-good-design-5-examples-we-can-learn-frombad-design-vs-good-design-5-examples-we-can-learn-from-130706

Spolsky, J. 2001. User Interface Design for Programmers. Apress.

Unity 3D Documentation. 2018. Canvas. https://unity3d.com/learn/tutorials/topics/user-interface-ui/ui-canvas?playlist=17111

Unity 3D Documentation. 2018. Render Mode. https://unity3d.com/learn/tutorials/topics/user-interface-ui/ui-canvas?playlist=17111

Unity 3D Documentation. 2018. RectTransform. https://unity3d.com/learn/tutorials/modules/beginner/ui/rect-transform?playlist=17111

Unreal Engine Documentation. 2018. UMG UI Designer. https://docs.unrealengine.com/en-us/Engine/UMG

Unreal Engine Documentation. 2018. Widget Bluprints. https://docs.unrealengine.com/en-US/Engine/UMG/UserGuide/WidgetBlueprints

Wikipedia. 2018. Adobe Flash. https://en.wikipedia.org/wiki/Adobe_Flash

Wikipedia. 2018. Adventure Game Studio. https://en.wikipedia.org/wiki/Adventure_Game_Studio

Wikipedia. 2018. Amazon Lumberyard. https://en.wikipedia.org/wiki/Amazon_Lumberyard

Wikipedia. 2018. CryEngine. https://en.wikipedia.org/wiki/CryEngine
Wikipedia. 2018. GameMaker Studio. https://en.wikipedia.org/wiki/GameMaker_Studio
Wikipedia. 2018. Unity (game engine). https://en.wikipedia.org/wiki/Unity_(game_engine)
Wikipedia. 2018. Unreal Engine. https://en.wikipedia.org/wiki/Unreal_Engine